

The IMA Volumes in Mathematics and its Applications

Jon Lee
Sven Leyffer *Editors*

Mixed Integer Nonlinear Programming

 Springer

The IMA Volumes in Mathematics
and its Applications
Volume 154

For further volumes:
<http://www.springer.com/series/811>

Institute for Mathematics and its Applications (IMA)

The **Institute for Mathematics and its Applications** was established by a grant from the National Science Foundation to the University of Minnesota in 1982. The primary mission of the IMA is to foster research of a truly interdisciplinary nature, establishing links between mathematics of the highest caliber and important scientific and technological problems from other disciplines and industries. To this end, the IMA organizes a wide variety of programs, ranging from short intense workshops in areas of exceptional interest and opportunity to extensive thematic programs lasting a year. IMA Volumes are used to communicate results of these programs that we believe are of particular value to the broader scientific community.

The full list of IMA books can be found at the Web site of the Institute for Mathematics and its Applications:

<http://www.ima.umn.edu/springer/volumes.html>.

Presentation materials from the IMA talks are available at

<http://www.ima.umn.edu/talks/>.

Video library is at

<http://www.ima.umn.edu/videos/>.

Fadil Santosa, Director of the IMA

* * * * *

IMA ANNUAL PROGRAMS

1982–1983	Statistical and Continuum Approaches to Phase Transition
1983–1984	Mathematical Models for the Economics of Decentralized Resource Allocation
1984–1985	Continuum Physics and Partial Differential Equations
1985–1986	Stochastic Differential Equations and Their Applications
1986–1987	Scientific Computation
1987–1988	Applied Combinatorics
1988–1989	Nonlinear Waves
1989–1990	Dynamical Systems and Their Applications
1990–1991	Phase Transitions and Free Boundaries
1991–1992	Applied Linear Algebra

Continued at the back

Jon Lee • Sven Leyffer
Editors

Mixed Integer Nonlinear Programming

Editors

Jon Lee
Industrial and Operations Engineering
University of Michigan
1205 Beal Avenue
Ann Arbor, Michigan 48109
USA

Sven Leyffer
Mathematics and Computer Science
Argonne National Laboratory
Argonne, Illinois 60439
USA

ISSN 0940-6573

ISBN 978-1-4614-1926-6 e-ISBN 978-1-4614-1927-3

DOI 10.1007/978-1-4614-1927-3

Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2011942482

Mathematics Subject Classification (2010): 05C25, 20B25, 49J15, 49M15, 49M37, 49N90, 65K05, 90C10, 90C11, 90C22, 90C25, 90C26, 90C27, 90C30, 90C35, 90C51, 90C55, 90C57, 90C60, 90C90, 93C95

© Springer Science+Business Media, LLC 2012

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

FOREWORD

This IMA Volume in Mathematics and its Applications

MIXED INTEGER NONLINEAR PROGRAMMING

contains expository and research papers based on a highly successful IMA Hot Topics Workshop “Mixed-Integer Nonlinear Optimization: Algorithmic Advances and Applications”. We are grateful to all the participants for making this occasion a very productive and stimulating one.

We would like to thank Jon Lee (Industrial and Operations Engineering, University of Michigan) and Sven Leyffer (Mathematics and Computer Science, Argonne National Laboratory) for their superb role as program organizers and editors of this volume.

We take this opportunity to thank the National Science Foundation for its support of the IMA.

Series Editors

Fadil Santosa, Director of the IMA

Markus Keel, Deputy Director of the IMA

PREFACE

Many engineering, operations, and scientific applications include a mixture of discrete and continuous decision variables and nonlinear relationships involving the decision variables that have a pronounced effect on the set of feasible and optimal solutions. Mixed-integer nonlinear programming (MINLP) problems combine the numerical difficulties of handling nonlinear functions with the challenge of optimizing in the context of nonconvex functions and discrete variables. MINLP is one of the most flexible modeling paradigms available for optimization; but because its scope is so broad, in the most general cases it is hopelessly intractable. Nonetheless, an expanding body of researchers and practitioners — including chemical engineers, operations researchers, industrial engineers, mechanical engineers, economists, statisticians, computer scientists, operations managers, and mathematical programmers — are interested in solving large-scale MINLP instances.

Of course, the wealth of applications that can be accurately modeled by using MINLP is not yet matched by the capability of available optimization solvers. Yet, the two key components of MINLP — mixed-integer linear programming (MILP) and nonlinear programming (NLP) — have experienced tremendous progress over the past 15 years. By cleverly incorporating many theoretical advances in MILP research, powerful academic, open-source, and commercial solvers have paved the way for MILP to emerge as a viable, widely used decision-making tool. Similarly, new paradigms and better theoretical understanding have created faster and more reliable NLP solvers that work well, even under adverse conditions such as failure of constraint qualifications.

In the fall of 2008, a Hot-Topics Workshop on MINLP was organized at the IMA, with the goal of synthesizing these advances and inspiring new ideas in order to transform MINLP. The workshop attracted more than 75 attendees, over 20 talks, and over 20 posters. The present volume collects 22 invited articles, organized into nine sections on the diverse aspects of MINLP. The volume includes survey articles, new research material, and novel applications of MINLP.

In its most general and abstract form, a MINLP can be expressed as

$$\underset{x}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad x \in \mathcal{F}, \quad (1)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a function and the feasible set \mathcal{F} contains both nonlinear and discrete structure. We note that we do not generally assume smoothness of f or convexity of the functions involved. Different realizations of the objective function f and the feasible set \mathcal{F} give rise to key classes of MINLPs addressed by papers in this collection.

Part I. Convex MINLP. Even though mixed-integer optimization problems are nonconvex as a result of the presence of discrete variables, the term *convex MINLP* is commonly used to refer to a class of MINLPs for which a convex program results when any explicit restrictions of discreteness on variables are relaxed (i.e., removed). In its simplest definition, for a convex MINLP, we may assume that the objective function f in (1) is a convex function and that the feasible set \mathcal{F} is described by a set of convex nonlinear functions, $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$, and a set of indices, $\mathcal{I} \subset \{1, \dots, n\}$, of integer variables:

$$\mathcal{F} = \{x \in \mathbb{R}^n \mid c(x) \leq 0, \text{ and } x_i \in \mathbb{Z}, \forall i \in \mathcal{I}\}. \quad (2)$$

Typically, we also demand some smoothness of the functions involved. Sometimes it is useful to expand the definition of convex MINLP to simply require that the functions be convex *on the feasible region*. Besides problems that can be directly modeled as convex MINLPs, the subject has relevance to methods that create convex MINLP subproblems.

Algorithms and software for convex mixed-integer nonlinear programs (*P. Bonami, M. Kilinç, and J. Linderoth*) discusses the state of the art for algorithms and software aimed at convex MINLPs. Important elements of successful methods include a tree search (to handle the discrete variables), NLP subproblems to tighten linearizations, and MILP master problems to collect and exploit the linearizations.

A special type of convex constraint is a second-order cone constraint: $\|y\|_2 \leq z$, where y is vector variable and z is a scalar variable. *Subgradient-based outer approximation for mixed-integer second-order cone programming* (*S. Drewes and S. Ulbrich*) demonstrates how such constraints can be handled by using outer-approximation techniques. A main difficulty, which the authors address using subgradients, is that at the point $(y, z) = (0, 0)$, the function $\|y\|_2$ is not differentiable.

Many convex MINLPs have “off/on” decisions that force a continuous variable either to be 0 or to be in a convex set. *Perspective reformulation and applications* (*O. Günlük and J. Linderoth*) describes an effective reformulation technique that is applicable to such situations. The perspective $g(x, t) = tc(x/t)$ of a convex function $c(x)$ is itself convex, and this property can be used to construct tight reformulations. The perspective reformulation is closely related to the subject of the next section: disjunctive programming.

Part II. Disjunctive programming. Disjunctive programs involve continuous variable together with Boolean variables which model logical propositions directly rather than by means of an algebraic formulation.

Generalized disjunctive programming: A framework for formulation and alternative algorithms for MINLP optimization (*I.E. Grossmann and J.P. Ruiz*) addresses generalized disjunctive programs (GDPs), which are MINLPs that involve general disjunctions and nonlinear terms. GDPs can

be formulated as MINLPs either through the “big-M” formulation, or by using the perspective of the nonlinear functions. The authors describe two approaches: disjunctive branch-and-bound, which branches on the disjunctions, and logic-based outer approximation, which constructs a disjunctive MILP master problem.

Under the assumption that the problem functions are factorable (i.e., the functions can be computed in a finite number of simple steps by using unary and binary operators), a MINLP can be reformulated as an equivalent MINLP where the only nonlinear constraints are equations involving two or three variables. The paper *Disjunctive cuts for nonconvex MINLP* (*P. Belotti*) describes a procedure for generating disjunctive cuts. First, spatial branching is performed on an original problem variable. Next, bound reduction is applied to the two resulting relaxations, and linear relaxations are created from a small number of outer approximations of each nonlinear expression. Then a cut-generation LP is used to produce a new cut.

Part III. Nonlinear programming. For several important and practical approaches to solving MINLPs, the most important part is the fast and accurate solution of NLP subproblems. NLPs arise both as nodes in branch-and-bound trees and as subproblems for fixed integer or Boolean variables. The papers in this section discuss two complementary techniques for solving NLPs: active-set methods in the form of sequential quadratic programming (SQP) methods and interior-point methods (IPMs).

Sequential quadratic programming methods (*P.E. Gill and E. Wong*) is a survey of a key NLP approach, sequential quadratic programming (SQP), that is especially relevant to MINLP. SQP methods solve NLPs by a sequence of quadratic programming approximations and are particularly well-suited to warm starts and re-solves that occur in MINLP.

IPMs are an alternative to SQP methods. However, standard IPMs can stall if started near a solution, or even fail on infeasible NLPs, making them less suitable for MINLP. *Using interior-point methods within an outer approximation framework for mixed-integer nonlinear programming* (*H. Y. Benson*) suggests a primal-dual regularization that penalizes the constraints and bounds the slack variables to overcome the difficulties caused by warm starts and infeasible subproblems.

Part IV. Expression graphs. Expression graphs are a convenient way to represent functions. An expression graph is a directed graph in which each node represents an arithmetic operation, incoming edges represent operations, and outgoing edges represent the result of the operation. Expression graphs can be manipulated to obtain derivative information, perform problem simplifications through presolve operations, or obtain relaxations of nonconvex constraints.

Using expression graphs in optimization algorithms (*D.M. Gay*) discusses how expression graphs allow gradients and Hessians to be computed

efficiently by exploiting group partial separability. In addition, the author describes how expression graphs can be used to tighten bounds on variables to provide tighter outer approximations of nonconvex expressions, detect convexity (e.g., for quadratic constraints), and propagate constraints.

Symmetry arises in many MINLP formulations and can mean that a problem or subproblem may have many symmetric optima or near optima, resulting in large search trees and inefficient pruning. *Symmetry in mathematical programming (L. Liberti)* describes how the symmetry group of a MINLP can be detected by parsing the expression graph. Once the symmetry group is known, we can add symmetry-breaking constraints or employ special branching schemes such as orbital branching that mitigate the adverse effects of symmetry.

Part V. Convexification and linearization. A popular and classical approach for handling nonconvex functions is to approximate them by using piecewise-linear functions. This approach requires the addition of binary variables that model the piecewise approximation. The advantage of such an approach is that advanced MILP techniques can be applied. The disadvantage of the approach is that the approximations are not exact and that it suffers from the curse of dimensionality.

Using piecewise linear functions for solving MINLPs (B. Geißler, A. Martin, A. Morsi, and L. Schewe) details how to carry out piecewise-linear approximation for MINLP. The authors review two formulations of piecewise linearization: the convex combination technique and the incremental technique. They introduce a piecewise-polyhedral outer-approximation algorithm based on rigorous error estimates, and they demonstrate computational success on water network and gas network problems.

A global-optimization algorithm for mixed-integer nonlinear programs having separable nonconvexity (C. D'Ambrosio, J. Lee, and A. Wächter) introduces a method for MINLPs that have all of their nonconvexity in separable form. The approach aims to retain and exploit existing convexity in the formulation.

Global optimization of mixed-integer signomial programming problems (A. Lundell and T. Westerlund) describes a global optimization algorithm for MINLPs containing signomial functions. The method obtains a convex relaxation through reformulations, by using single-variable transformations in concert with piecewise-linear approximations of the inverse transformations.

Part VI. Mixed-integer quadratically-constrained optimization. In seeking a more structured setting than general MINLP, but with considerably more modeling power than is afforded by MILP, one naturally considers mixed-integer models with quadratic functions, namely, MIQCPs. Such models are NP-hard, but they have enough structure that can be exploited in order to gain computational advantages over treating such problems as general MINLPs.

The MILP road to MIQCP (*S. Burer and A. Saxena*) surveys results in mixed-integer quadratically constrained programming. Strong convex relaxations and valid inequalities are the basis of efficient, practical techniques for global optimization. Some of the relaxations and inequalities are derived from the algebraic formulation, while others are based on disjunctive programming. Much of the inspiration derives from MILP methodology.

Linear programming relaxations of quadratically-constrained quadratic programs (*A. Qualizza, P. Belotti, and F. Margot*) investigates the use of LP tools for approximately solving semidefinite programming (SDP) relaxations of quadratically-constrained quadratic programs. The authors present classes of valid linear inequalities based on spectral decomposition, together with computational results.

Extending a CIP framework to solve MIQCPs (*T. Berthold, S. Heinz, and S. Vigerske*) discusses how to build a solver for MIQCPs by extending a framework for constraint integer programming (CIP). The advantage of this approach is that we can utilize the full power of advanced MILP and constraint programming technologies. For relaxation, the approach employs an outer approximation generated by linearization of convex constraints and linear underestimation of nonconvex constraints. Reformulation, separation, and propagation techniques are used to handle the quadratic constraints efficiently. The authors implemented these methods in the branch-cut-and-price framework SCIP.

Part VII. Combinatorial optimization. Because of the success of MILP methods and because of beautiful and algorithmically important results from polyhedral combinatorics, nonlinear functions and formulations have not been heavily investigated for combinatorial optimization problems. With improvements in software for general NLP, SDP, and MINLP, however, researchers are now investing considerable effort in trying to exploit these gains for combinatorial-optimization problems.

Computation with polynomial equations and inequalities arising in combinatorial optimization (*J.A. De Loera, P.N. Malkin, and P.A. Parrilo*) discusses how the algebra of multivariate polynomials can be used to create large-scale linear algebra or semidefinite-programming relaxations of many kinds of combinatorial feasibility and optimization problems.

Matrix relaxations in combinatorial optimization (*F. Rendl*) discusses the use of SDP as a modeling tool in combinatorial optimization. The main techniques to get matrix relaxations of combinatorial-optimization problems are presented. Semidefiniteness constraints lead to tractable relaxations, while constraints that matrices be completely positive or copositive do not. This survey illustrates the enormous power and potential of matrix relaxations.

A polytope for a product of real linear functions in 0/1 variables (*O. Günlük, J. Lee, and J. Leung*) uses polyhedral methods to give a tight

formulation for the convex hull of a product of two linear functions in 0/1 variables. As an example, by writing a pair of general integer variables in binary expansion, the authors have a technique for linearizing their product.

Part VIII. Complexity. General MINLP is incomputable, independent of conjectures such as P \neq NP. From the point of view of complexity theory, however, considerable room exists for negative results (e.g., incomputability, intractability and inapproximability results) and positive results (e.g., polynomial-time algorithms and approximations schemes) for restricted classes of MINLPs.

On the complexity of nonlinear mixed-integer optimization (M. Köppe) is a survey on the computational complexity of MINLP. It includes incomputability results that arise from number theory and logic, fully polynomial-time approximation schemes in fixed dimension, and polynomial-time algorithms for special cases.

Theory and applications of n-fold integer programming (S. Onn) is an overview of the theory of n-fold integer programming, which enables the polynomial-time solution of fundamental linear and nonlinear integer programming problems in variable dimension. This framework yields polynomial-time algorithms in several application areas, including multi-commodity flows and privacy in statistical databases.

Part IX. Applications. A wide range of applications of MINLP exist. This section focuses on two new application domains.

MINLP application for ACH interiors restructuring (E. Klampfl and Y. Fradkin) describes a very large-scale application of MINLP developed by the Ford Motor Company. The MINLP models the re-engineering of 42 product lines over 26 manufacturing processes and 50 potential supplier sites. The resulting MINLP model has 350,000 variables (17,000 binary) and 1.6 million constraints and is well beyond the size that state-of-the-art MINLP solvers can handle. The authors develop a piecewise-linearization scheme for the objective and a decomposition technique that decouples the problem into two coupled MILPs that are solved iteratively.

A benchmark library of mixed-integer optimal control problems (S. Sager) describes a challenging new class of MINLPs. These are optimal control problems, involving differential-algebraic equation constraints and integrality restrictions on the controls, such as gear ratios. The authors describe 12 models from a range of applications, including biology, industrial engineering, trajectory optimization, and process control.

Acknowledgments. We gratefully acknowledge the generous financial support from the IMA that made this workshop possible, as well as financial support from IBM. This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. Special

thanks are due to Fadil Santosa, Chun Liu, Patricia Brick, Dzung Nguyen, Holly Pinkerton, and Eve Marofsky from the IMA, who made the organization of the workshop and the publication of this special volume such an easy and enjoyable affair.

Jon Lee

University of Michigan

Sven Leyffer

Argonne National Laboratory

CONTENTS

Foreword	v
Preface	vii

PART I: CONVEX MINLP

Algorithms and software for convex mixed integer nonlinear programs	1
<i>Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth</i>	
Subgradient based outer approximation for mixed integer second order cone programming.....	41
<i>Sarah Drewes and Stefan Ulbrich</i>	
Perspective reformulation and applications.....	61
<i>Oktay Günlük and Jeff Linderoth</i>	

PART II: DISJUNCTIVE PROGRAMMING

Generalized disjunctive programming: A framework for formulation and alternative algorithms for MINLP optimization.....	93
<i>Ignacio E. Grossmann and Juan P. Ruiz</i>	
Disjunctive cuts for nonconvex MINLP	117
<i>Pietro Belotti</i>	

PART III: NONLINEAR PROGRAMMING

Sequential quadratic programming methods	147
<i>Philip E. Gill and Elizabeth Wong</i>	
Using interior-point methods within an outer approximation framework for mixed integer nonlinear programming	225
<i>Hande Y. Benson</i>	

PART IV: EXPRESSION GRAPHS

Using expression graphs in optimization algorithms	247
<i>David M. Gay</i>	
Symmetry in mathematical programming.....	263
<i>Leo Liberti</i>	

PART V: CONVEXIFICATION AND LINEARIZATION

Using piecewise linear functions for solving MINLPs	287
<i>Björn Geißler, Alexander Martin, Antonio Morsi, and Lars Schewe</i>	
An algorithmic framework for MINLP with separable non-convexity	315
<i>Claudia D'Ambrosio, Jon Lee, and Andreas Wächter</i>	
Global optimization of mixed-integer signomial programming problems	349
<i>Andreas Lundell and Tapio Westerlund</i>	

PART VI: MIXED-INTEGER QUADRATICALLY
CONSTRAINED OPTIMIZATION

The MILP road to MIQCP	373
<i>Samuel Burer and Anureet Saxena</i>	
Linear programming relaxations of quadratically constrained quadratic programs	407
<i>Andrea Qualizza, Pietro Belotti, and François Margot</i>	
Extending a CIP framework to solve MIQCPs	427
<i>Timo Berthold, Stefan Heinz, and Stefan Vigerske</i>	

PART VII: COMBINATORIAL OPTIMIZATION

Computation with polynomial equations and inequalities arising in combinatorial optimization	447
<i>Jesus A. De Loera, Peter N. Malkin, and Pablo A. Parrilo</i>	
Matrix relaxations in combinatorial optimization	483
<i>Franz Rendl</i>	
A polytope for a product of real linear functions in 0/1 variables	513
<i>Oktay Günlük, Jon Lee, and Janny Leung</i>	

PART VIII: COMPLEXITY

On the complexity of nonlinear mixed-integer optimization	533
<i>Matthias Köppe</i>	
Theory and applications of n -fold integer programming	559
<i>Shmuel Onn</i>	

PART IX: APPLICATIONS

MINLP Application for ACH interiors restructuring	597
<i>Erica Klampfl and Yakov Fradkin</i>	
A benchmark library of mixed-integer optimal control problems.....	631
<i>Sebastian Sager</i>	
List of Hot Topics participants	671

PART I:

CONVEX MINLP

ALGORITHMS AND SOFTWARE FOR CONVEX MIXED INTEGER NONLINEAR PROGRAMS

PIERRE BONAMI*, MUSTAFA KILINÇ†, AND JEFF LINDEROTH‡

Abstract. This paper provides a survey of recent progress and software for solving convex Mixed Integer Nonlinear Programs (MINLP)s, where the objective and constraints are defined by convex functions and integrality restrictions are imposed on a subset of the decision variables. *Convex MINLPs* have received sustained attention in recent years. By exploiting analogies to well-known techniques for solving Mixed Integer Linear Programs and incorporating these techniques into software, significant improvements have been made in the ability to solve these problems.

Key words. Mixed Integer Nonlinear Programming; Branch and Bound.

1. Introduction. Mixed-Integer Nonlinear Programs (MINLP)s are optimization problems where some of the variables are constrained to take integer values and the objective function and feasible region of the problem are described by nonlinear functions. Such optimization problems arise in many real world applications. Integer variables are often required to model logical relationships, fixed charges, piecewise linear functions, disjunctive constraints and the non-divisibility of resources. Nonlinear functions are required to accurately reflect physical properties, covariance, and economies of scale.

In full generality, MINLPs form a particularly broad class of challenging optimization problems, as they combine the difficulty of optimizing over integer variables with the handling of nonlinear functions. Even if we restrict our model to contain only linear functions, MINLP reduces to a Mixed-Integer Linear Program (MILP), which is an NP-Hard problem [55]. On the other hand, if we restrict our model to have no integer variable but allow for general nonlinear functions in the objective or the constraints, then MINLP reduces to a Nonlinear Program (NLP) which is also known to be NP-Hard [90]. Combining both integrality and nonlinearity can lead to examples of MINLP that are undecidable [67].

*Laboratoire d’Informatique Fondamentale de Marseille, CNRS, Aix-Marseille Universités, Parc Scientifique et Technologique de Luminy, 163 avenue de Luminy - Case 901, F-13288 Marseille Cedex 9, France (pierre.bonami@lif.univ-mrs.fr). Supported by ANR grand BLAN06-1-138894.

†Department of Industrial and Systems Engineering, University of Wisconsin-Madison, 1513 University Ave., Madison, WI, 53706 (kilinc@wisc.edu).

‡Department of Industrial and Systems Engineering, University of Wisconsin-Madison, 1513 University Ave., Madison, WI 53706 (linderoth@wisc.edu). The work of the second and third authors is supported by the US Department of Energy under grants DE-FG02-08ER25861 and DE-FG02-09ER25869, and the National Science Foundation under grant CCF-0830153.

In this paper, we restrict ourselves to the subclass of MINLP where the objective function to minimize is convex, and the constraint functions are all convex and upper bounded. In these instances, when integrality is relaxed, the feasible set is convex. Convex MINLP is still NP-hard since it contains MILP as a special case. Nevertheless, it can be solved much more efficiently than general MINLP since the problem obtained by dropping the integrality requirements is a convex NLP for which there exist efficient algorithms. Further, the convexity of the objective function and feasible region can be used to design specialized algorithms.

There are many diverse and important applications of MINLPs. A small subset of these applications includes portfolio optimization [21, 68], block layout design in the manufacturing and service sectors [33, 98], network design with queuing delay constraints [27], integrated design and control of chemical processes [53], drinking water distribution systems security [73], minimizing the environmental impact of utility plants [46], and multi-period supply chain problems subject to probabilistic constraints [75].

Even though convex MINLP is NP-Hard, there are exact methods for its solution—methods that terminate with a guaranteed optimal solution or prove that no such solution exists. In this survey, our main focus is on such exact methods and their implementation.

In the last 40 years, at least five different algorithms have been proposed for solving convex MINLP to optimality. In 1965, Dakin remarked that the branch-and-bound method did not require linearity and could be applied to convex MINLP. In the early 70's, Geoffrion [56] generalized Benders decomposition to make an exact algorithm for convex MINLP. In the 80's, Gupta and Ravindran studied the application of branch and bound [62]. At the same time, Duran and Grossmann [43] introduced the Outer Approximation decomposition algorithm. This latter algorithm was subsequently improved in the 90's by Fletcher and Leyffer [51] and also adapted to the branch-and-cut framework by Quesada and Grossmann [96]. In the same period, a related method called the Extended Cutting Plane method was proposed by Westerlund and Pettersson [111]. Section 3 of this paper will be devoted to reviewing in more detail all of these methods.

Two main ingredients of the above mentioned algorithms are solving MILP and solving NLP. In the last decades, there have been enormous advances in our ability to solve these two important subproblems of convex MINLP.

We refer the reader to [100, 92] and [113] for in-depth analysis of the theory of MILP. The advances in the theory of solving MILP have led to the implementation of solvers both commercial and open-source which are now routinely used to solve many industrial problems of large size. Bixby and Rothberg [22] demonstrate that advances in algorithmic technology alone have resulted in MILP instances solving more than 300 times faster than a decade ago. There are effective, robust commercial MILP solvers

such as CPLEX [66], XPRESS-MP [47], and Gurobi [63]. Linderoth and Ralphs [82] give a survey of noncommercial software for MILP.

There has also been steady progress over the past 30 years in the development and successful implementation of algorithms for NLPs. We refer the reader to [12] and [94] for a detailed recital of nonlinear programming techniques. Theoretical developments have led to successful implementations in software such as SNOPT [57], filterSQP [52], CONOPT [42], IPOPT [107], LOQO [103], and KNITRO [32]. Waltz [108] states that the size of instance solvable by NLP is growing by nearly an order of magnitude a decade.

Of course, solution algorithms for convex MINLP have benefit from the technological progress made in solving MILP and NLP. However, in the realm of MINLP, the progress has been far more modest, and the dimension of solvable convex MINLP by current solvers is small when compared to MILPs and NLPs. In this work, our goal is to give a brief introduction to the techniques which are in state-of-the-art solvers for convex MINLPs. We survey basic theory as well as recent advances that have made their way into software. We also attempt to make a fair comparison of all algorithmic approaches and their implementations.

The remainder of the paper can be outlined as follows. A precise description of a MINLP and algorithmic building blocks for solving MINLPs are given in Section 2. Section 3 outlines five different solution techniques. In Section 4, we describe in more detail some advanced techniques implemented in the latest generation of solvers. Section 5 contains descriptions of several state-of-the-art solvers that implement the different solution techniques presented. Finally, in Section 6 we present a short computational comparison of those software packages.

2. MINLP. The focus of this section is to mathematically define a MINLP and to describe important special cases. Basic elements of algorithms and subproblems related to MINLP are also introduced.

2.1. MINLP problem classes. A Mixed Integer Nonlinear Program may be expressed in algebraic form as follows:

$$\begin{aligned} z_{\text{MINLP}} = & \text{minimize } f(x) \\ \text{subject to } & g_j(x) \leq 0 \quad \forall j \in J, \\ & x \in X, \quad x_I \in \mathbb{Z}^{|I|}, \end{aligned} \tag{MINLP}$$

where X is a polyhedral subset of \mathbb{R}^n (e.g. $X = \{x \mid x \in \mathbb{R}_+^n, Ax \leq b\}$). The functions $f : X \rightarrow \mathbb{R}$ and $g_j : X \rightarrow \mathbb{R}$ are sufficiently smooth functions. The algorithms presented here only require continuously differentiable functions, but in general algorithms for solving continuous relaxations converge much faster if functions are twice-continuously differentiable. The set J is the index set of nonlinear constraints, I is the index set of discrete variables and C is the index set of continuous variables, so $I \cup C = \{1, \dots, n\}$.

For convenience, we assume that the set X is bounded; in particular some finite lower bounds L_I and upper bounds U_I on the values of the integer variables are known. In most applications, discrete variables are restricted to 0-1 values, i.e., $x_i \in \{0, 1\} \forall i \in I$. In this survey, we focus on the case where the functions f and g_j are convex. Thus, by relaxing the integrality constraint on x , a convex program, minimization of a convex function over a convex set, is formed. We will call such problems *convex MINLPs*. From now on, unless stated, we will refer convex MINLPs as MINLPs.

There are a number of important special cases of MINLP. If $f(x) = x^T Qx + d^T x + h$, is a (convex) quadratic function of x , and there are only linear constraints on the problem ($J = \emptyset$), the problem is known as a mixed integer quadratic program (MIQP). If both $f(x)$ and $g_j(x)$ are quadratic functions of x for each $j \in J$, the problem is known as a mixed integer quadratically constrained program (MIQCP). Significant work has been devoted to these important special cases [87, 29, 21].

If the objective function is linear, and all nonlinear constraints have the form $g_j(x) = \|Ax + b\|_2 - c^T x - d$, then the problem is a Mixed Integer Second-Order Cone Program (MISOCP). Through a well-known transformation, MIQCP can be transformed into a MISOCP. In fact, many different types of sets defined by nonlinear constraints are representable via second-order cone inequalities. Discussion of these transformations is out of the scope of this work, but the interested reader may consult [15]. Relatively recently, commercial software packages such as CPLEX [66], XPRESS-MP [47], and Mosek [88] have all been augmented to include specialized algorithms for solving these important special cases of convex MINLPs. In what follows, we focus on general convex MINLP and software available for its solution.

2.2. Basic elements of MINLP methods. The basic concept underlying algorithms for solving (MINLP) is to generate and refine bounds on its optimal solution value. Lower bounds are generated by solving a relaxation of (MINLP), and upper bounds are provided by the value of a feasible solution to (MINLP). Algorithms differ in the manner in which these bounds are generated and the sequence of subproblems that are solved to generate these bounds. However, algorithms share many basic common elements, which are described next.

Linearizations: Since the objective function of (MINLP) may be non-linear, its optimal solution may occur at a point that is interior to the convex hull of its set of feasible solutions. It is simple to transform the instance to have a linear objective function by introducing an auxiliary variable η and moving the original objective function into the constraints. Specifically, (MINLP) may be equivalently stated as

$$\begin{aligned}
z_{\text{MINLP}} = & \text{minimize} && \eta \\
& \text{subject to} && f(x) \leq \eta \\
& && g_j(x) \leq 0 \quad \forall j \in J, \\
& && x \in X, \quad x_I \in \mathbb{Z}^{|I|}.
\end{aligned} \tag{MINLP-1}$$

Many algorithms rely on linear relaxations of (MINLP), obtained by linearizing the objective and constraint functions at a given point \hat{x} . Since f and g_j are convex and differentiable, the inequalities

$$f(\hat{x}) + \nabla f(\hat{x})^T(x - \hat{x}) \leq f(x),$$

$$g_j(\hat{x}) + \nabla g_j(\hat{x})^T(x - \hat{x}) \leq g_j(x),$$

are valid for all $j \in J$ and $\hat{x} \in \mathbb{R}^n$. Since $f(x) \leq \eta$ and $g_j(x) \leq 0$, then the linear inequalities

$$f(\hat{x}) + \nabla f(\hat{x})^T(x - \hat{x}) \leq \eta, \tag{2.1}$$

$$g_j(\hat{x}) + \nabla g_j(\hat{x})^T(x - \hat{x}) \leq 0 \tag{2.2}$$

are valid for (MINLP-1). Linearizations of $g_j(x)$ outer approximate the feasible region, and linearizations of $f(x)$ underestimate the objective function. We often refer to (2.1)–(2.2) as outer approximation constraints.

Subproblems: One important subproblem used by a variety of algorithms for (MINLP) is formed by relaxing the integrity requirements and restricting the bounds on the integer variables. Given bounds $(l_I, u_I) = \{(\ell_i, u_i) \mid \forall i \in I\}$, the *NLP relaxation* of (MINLP) is

$$\begin{aligned}
z_{\text{NLPR}(l,u)} = & \text{minimize} && f(x) \\
& \text{subject to} && g_j(x) \leq 0 \quad \forall j \in J, \\
& && x \in X; \quad l_I \leq x_I \leq u_I.
\end{aligned} \tag{\text{NLPR}(l_I, u_I)}$$

The value $z_{\text{NLPR}(l,u)}$ is a lower bound on the value of z_{MINLP} that can be obtained in the subset of the feasible region of (MINLP) where the bounds $\ell_I \leq x_I \leq u_I$ are imposed. Specifically, if (l_I, u_I) are the lower and upper bounds (L_I, U_I) for the original instance, then $z_{\text{NLPR}(L_I, U_I)}$ provides a lower bound on z_{MINLP} .

In the special case that all of the integer variables are fixed ($l_I = u_I = \hat{x}_I$), the *fixed NLP subproblem* is formed:

$$\begin{aligned}
z_{\text{NLP}(\hat{x}_I)} = & \text{minimize} && f(x) \\
& \text{subject to} && g_j(x) \leq 0, \quad \forall j \in J \\
& && x \in X; \quad x_I = \hat{x}_I.
\end{aligned} \tag{\text{NLP}(\hat{x}_I)}$$

If $\hat{x}_I \in \mathbb{Z}^{|I|}$ and (NLP(\hat{x}_I)) has a feasible solution, the value $z_{\text{NLP}(\hat{x}_I)}$ provides an upper bound to the problem (MINLP). If (NLP(\hat{x}_I)) is infeasible,

NLP software typically will deduce infeasibility by solving an associated feasibility subproblem. One choice of feasibility subproblem employed by NLP solvers is

$$\begin{aligned} z_{\text{NLPF}(\hat{x}_I)} = \text{minimize} \quad & \sum_{j=1}^m w_j g_j(x)^+ \\ \text{s.t. } & x \in X, \quad x_I = \hat{x}_I, \end{aligned} \quad (\text{NLPF}(\hat{x}_I))$$

where $g_j(x)^+ = \max\{0, g_j(x)\}$ measures the violation of the nonlinear constraints and $w_j \geq 0$. Since when $\text{NLP}(\hat{x}_I)$ is infeasible NLP solvers will return the solution to $\text{NLPF}(\hat{x}_I)$, we will often say, by abuse of terminology, that $\text{NLP}(\hat{x}_I)$ is solved and its solution \bar{x} is optimal or minimally infeasible, meaning that it is the optimal solution to $\text{NLPF}(\hat{x}_I)$.

3. Algorithms for convex MINLP. With elements of algorithms defined, attention can be turned to describing common algorithms for solving MINLPs. The algorithms share many general characteristics with the well-known branch-and-bound or branch-and-cut methods for solving MILPs.

3.1. NLP-Based Branch and Bound. Branch and bound is a divide-and-conquer method. The dividing (branching) is done by partitioning the set of feasible solutions into smaller and smaller subsets. The conquering (fathoming) is done by bounding the value of the best feasible solution in the subset and discarding the subset if its bound indicates that it cannot contain an optimal solution.

Branch and bound was first applied to MILP by Land and Doig [74]. The method (and its enhancements such as branch and cut) remain the workhorse for all of the most successful MILP software. Dakin [38] realized that this method does not require linearity of the problem. Gupta and Ravindran [62] suggested an implementation of the branch-and-bound method for convex MINLPs and investigated different search strategies. Other early works related to NLP-Based Branch and Bound (NLP-BB for short) for convex MINLP include [91], [28], and [78].

In NLP-BB, the lower bounds come from solving the subproblems $(\text{NLPR}(l_I, u_I))$. Initially, the bounds (L_I, U_I) (the lower and upper bounds on the integer variables in (MINLP)) are used, so the algorithm is initialized with a continuous relaxation whose solution value provides a lower bound on z_{MINLP} . The variable bounds are successively refined until the subregion can be fathomed. Continuing in this manner yields a tree \mathcal{L} of subproblems. A node N of the search tree is characterized by the bounds enforced on its integer variables: $N \stackrel{\text{def}}{=} (l_I, u_I)$. Lower and upper bounds on the optimal solution value $z_L \leq z_{\text{MINLP}} \leq z_U$ are updated through the course of the algorithm. Algorithm 1 gives pseudocode for the NLP-BB algorithm for solving (MINLP).

Algorithm 1 The NLP-Based Branch-and-Bound algorithm

0. **Initialize.**
 $\mathcal{L} \leftarrow \{(L_I, U_I)\}$. $z_U = \infty$. $x^* \leftarrow \text{NONE}$.
 1. **Terminate?**
Is $\mathcal{L} = \emptyset$? If so, the solution x^* is optimal.
 2. **Select.**
Choose and delete a problem $N^i = (l_I^i, u_I^i)$ from \mathcal{L} .
 3. **Evaluate.**
Solve $\text{NLPR}(l_I^i, u_I^i)$. If the problem is infeasible, go to step 1, else let $z_{\text{NLPR}(l_I^i, u_I^i)}^*$ be its optimal objective function value and \hat{x}^i be its optimal solution.
 4. **Prune.**
If $z_{\text{NLPR}(l_I^i, u_I^i)}^* \geq z_U$, go to step 1. If \hat{x}^i is fractional, go to step 5, else let $z_U \leftarrow z_{\text{NLPR}(l_I^i, u_I^i)}^*$, $x^* \leftarrow \hat{x}^i$, and delete from \mathcal{L} all problems with $z_L^j \geq z_U$. Go to step 1.
 5. **Divide.**
Divide the feasible region of N^i into a number of smaller feasible subregions, creating nodes $N^{i_1}, N^{i_2}, \dots, N^{i_k}$. For each $j = 1, 2, \dots, k$, let $z_L^{i_j} \leftarrow z_{\text{NLPR}(l_I^i, u_I^i)}^*$ and add the problem N^{i_j} to \mathcal{L} . Go to 1.
-

As described in step 4 of Algorithm 1, if $\text{NLPR}(l_I^i, u_I^i)$ yields an integral solution (a solution where all discrete variables take integer values), then $z_{\text{NLPR}(l_I^i, u_I^i)}^*$ gives an upper bound for MINLP. Fathoming of nodes occurs when the lower bound for a subregion obtained by solving $\text{NLPR}(l_I^i, u_I^i)$ exceeds the current upper bound z_U , when the subproblem is infeasible, or when the subproblem provides a feasible integral solution. If none of these conditions is met, the node cannot be pruned and the subregion is divided to create new nodes. This **Divide** step of Algorithm 1 may be performed in many ways. In most successful implementations, the subregion is divided by *dichotomy branching*. Specifically, the feasible region of N^i is divided into subsets by changing bounds on one integer variable based on the solution \hat{x}^i to $\text{NLPR}(l_I^i, u_I^i)$. An index $j \in I$ such that $\hat{x}_j \notin \mathbb{Z}$ is chosen and two new children nodes are created by adding the bound $x_j \leq \lfloor \hat{x}_j \rfloor$ to one child and $x_j \geq \lceil \hat{x}_j \rceil$ to the other child. The tree search continues until all nodes are fathomed, at which point x^* is the optimal solution.

The description makes it clear that there are various choices to be made during the course of the algorithm. Namely, how do we select which subproblem to evaluate, and how do we divide the feasible region? A partial answer to these two questions will be provided in Sections 4.2 and 4.3. The NLP-Based Branch-and-Bound algorithm is implemented in solvers MINLP-BB [77], SBB [30], and Bonmin [24].

3.2. Outer Approximation. The Outer Approximation (OA) method for solving (MINLP) was first proposed by Duran and Grossmann [43]. The fundamental insight behind the algorithm is that (MINLP) is equivalent to a Mixed Integer *Linear* Program (MILP) of finite size. The MILP is constructed by taking linearizations of the objective and constraint functions about the solution to the subproblem $\text{NLP}(\hat{x}_I)$ or $\text{NLPF}(\hat{x}_I)$ for various choices of \hat{x}_I . Specifically, for each integer assignment $\hat{x}_I \in \text{Proj}_{x_I}(X) \cap \mathbb{Z}^{|I|}$ (where $\text{Proj}_{x_I}(X)$ denotes the projection of X onto the space of integer constrained variables), let $\bar{x} \in \arg \min \text{NLP}(\hat{x}_I)$ be an optimal solution to the NLP subproblem with integer variables fixed according to \hat{x}_I . If $\text{NLP}(\hat{x}_I)$ is not feasible, then let $\bar{x} \in \arg \min \text{NLPF}(\hat{x}_I)$ be an optimal solution to its corresponding feasibility problem. Since $\text{Proj}_{x_I}(X)$ is bounded by assumption, there are a finite number of subproblems $\text{NLP}(\hat{x}_I)$. For each of these subproblems, we choose one optimal solution, and let K be the (finite) set of these optimal solutions. Using these definitions, an outer-approximating MILP can be specified as

$$\begin{aligned} z_{\text{OA}} &= \min \quad \eta \\ \text{s.t. } \eta &\geq f(\bar{x}) + \nabla f(\bar{x})^T(x - \bar{x}) \quad \bar{x} \in K, \\ g_j(\bar{x}) + \nabla g_j(\bar{x})^T(x - \bar{x}) &\leq 0 \quad j \in J, \bar{x} \in K, \\ x &\in X, \quad x_I \in \mathbb{Z}^I. \end{aligned} \tag{MILP-OA}$$

The equivalence between (MINLP) and (MILP-OA) is specified in the following theorem:

THEOREM 3.1. [43, 51, 24] *If $X \neq \emptyset$, f and g are convex, continuously differentiable, and a constraint qualification holds for each $x^k \in K$ then $z_{\text{MINLP}} = z_{\text{OA}}$. All optimal solutions of (MINLP) are optimal solutions of (MILP-OA).*

From a practical point of view it is not relevant to try and formulate explicitly (MILP-OA) to solve (MINLP)—to explicitly build it, one would have first to enumerate all feasible assignments for the integer variables in X and solve the corresponding nonlinear programs $\text{NLP}(\hat{x}_I)$. The OA method uses an MILP relaxation ($\text{MP}(\mathcal{K})$) of (MINLP) that is built in a manner similar to (MILP-OA) but where linearizations are only taken at a subset \mathcal{K} of K :

$$\begin{aligned} z_{\text{MP}(\mathcal{K})} &= \min \quad \eta \\ \text{s.t. } \eta &\geq f(\bar{x}) + \nabla f(\bar{x})^T(x - \bar{x}) \quad \bar{x} \in \mathcal{K}, \\ g_j(\bar{x}) + \nabla g_j(\bar{x})^T(x - \bar{x}) &\leq 0 \quad j \in J, \bar{x} \in \mathcal{K}, \\ x &\in X, \quad x_I \in \mathbb{Z}^I. \end{aligned} \tag{\text{MP}(\mathcal{K})}$$

We call this problem the *OA-based reduced master problem*. The solution value of the reduced master problem ($\text{MP}(\mathcal{K})$), $z_{\text{MP}(\mathcal{K})}$, gives a lower bound to (MINLP), since $\mathcal{K} \subseteq K$. The OA method proceeds by iteratively

adding points to the set \mathcal{K} . Since function linearizations are accumulated as iterations proceed, the reduced master problem ($\text{MP}(\mathcal{K})$) yields a non-decreasing sequence of lower bounds.

OA typically starts by solving $(\text{NLPR}(L_I, U_I))$. Linearizations about the optimal solution to $(\text{NLPR}(l_I, u_I))$ are used to construct the first reduced master problem ($\text{MP}(\mathcal{K})$). Then, $(\text{MP}(\mathcal{K}))$ is solved to optimality to give an integer solution, \hat{x} . This integer solution is then used to construct the NLP subproblem $(\text{NLP}(\hat{x}_I))$. If $(\text{NLP}(\hat{x}_I))$ is feasible, linearizations about the optimal solution of $(\text{NLP}(\hat{x}_I))$ are added to the reduced master problem. These linearizations eliminate the current solution \hat{x} from the feasible region of $(\text{MP}(\mathcal{K}))$ unless \hat{x} is optimal for (MINLP). Also, the optimal solution value $z_{\text{NLP}(\hat{x}_I)}$ yields an upper bound to MINLP. If $(\text{NLP}(\hat{x}_I))$ is infeasible, the feasibility subproblem $(\text{NLPF}(\hat{x}_I))$ is solved and linearizations about the optimal solution of $(\text{NLPF}(\hat{x}_I))$ are added to the reduced master problem ($\text{MP}(\mathcal{K})$). The algorithm iterates until the lower and upper bounds are within a specified tolerance ϵ . Algorithm 2 gives pseudocode for the method. Theorem 3.1 guarantees that this algorithm cannot cycle and terminates in a finite number of steps.

Note that the reduced master problem need not be solved to optimality. In fact, given the upper bound UB and a tolerance ϵ , it is sufficient to generate any new $(\hat{\eta}, \hat{x})$ that is feasible to $(\text{MP}(\mathcal{K}))$, satisfies the integrality requirements, and for which $\eta \leq UB - \epsilon$. This can usually be achieved by setting a *cutoff value* in the MILP software to enforce the constraint $\eta \leq UB - \epsilon$. If a cutoff value is not used, then the infeasibility of $(\text{MP}(\mathcal{K}))$ implies the infeasibility of (MINLP). If a cutoff value is used, the OA iterations are terminated (Step 1 of Algorithm 2) when the OA master problem has no feasible solution. OA is implemented in the software packages DICOPT [60] and Bonmin [24].

3.3. Generalized Benders Decomposition. Benders Decomposition was introduced by Benders [16] for the problems that are linear in the “easy” variables, and nonlinear in the “complicating” variables. Geoffrion [56] introduced the Generalized Benders Decomposition (GBD) method for MINLP. The GBD method is very similar to the OA method, differing only in the definition of the MILP master problem. Specifically, instead of using linearizations for each nonlinear constraint, GBD uses duality theory to derive one single constraint that combines the linearizations derived from all the original problem constraints.

In particular, let \bar{x} be the optimal solution to $(\text{NLP}(\hat{x}_I))$ for a given integer assignment \hat{x}_I and $\bar{\mu} \geq 0$ be the corresponding optimal Lagrange multipliers. The following generalized Benders cut is valid for (MINLP)

$$\eta \geq f(\bar{x}) + (\nabla_I f(\bar{x}) + \nabla_I g(\bar{x})\bar{\mu})^T(x_I - \hat{x}_I). \quad (\text{BC}(\hat{x}))$$

Note that $\bar{x}_I = \hat{x}_I$, since the integer variables are fixed. In $(\text{BC}(\hat{x}))$, ∇_I refers to the gradients of functions f (or g) with respect to discrete vari-

Algorithm 2 The Outer Approximation algorithm.0. **Initialize.**

$z_U \leftarrow +\infty$. $z_L \leftarrow -\infty$. $x^* \leftarrow \text{NONE}$. Let \bar{x}^0 be the optimal solution of $(\text{NLPR}(L_I, U_I))$
 $\mathcal{K} \leftarrow \{\bar{x}^0\}$. Choose a convergence tolerance ϵ .

1. **Terminate?**

Is $z_U - z_L < \epsilon$ or $(\text{MP}(\mathcal{K}))$ infeasible? If so, x^* is ϵ -optimal.

2. **Lower Bound**

Let $z_{\text{MP}(\mathcal{K})}$ be the optimal value of $\text{MP}(\mathcal{K})$ and $(\hat{\eta}, \hat{x})$ its optimal solution.

$$z_L \leftarrow z_{\text{MP}(\mathcal{K})}$$

3. **NLP Solve**

Solve $(\text{NLP}(\hat{x}_I))$.

Let \bar{x}^i be the optimal (or minimally infeasible) solution.

4. **Upper Bound?**

Is \bar{x}^i feasible for (MINLP) and $f(\bar{x}^i) < z_U$? If so, $x^* \leftarrow \bar{x}^i$ and $z_U \leftarrow f(\bar{x}^i)$.

5. **Refine**

$$\mathcal{K} \leftarrow \mathcal{K} \cup \{\bar{x}^i\}$$

Go to 1.

ables. The inequality $(\text{BC}(\hat{x}))$ is derived by building a surrogate of the OA constraints using the multipliers $\bar{\mu}$ and simplifying the result using the Karush-Kuhn-Tucker conditions satisfied by \bar{x} (which in particular eliminates the continuous variables from the inequality).

If there is no feasible solution to $(\text{NLP}(\hat{x}_I))$, a feasibility cut can be obtained similarly by using the solution \bar{x} to $(\text{NLPF}(\hat{x}_I))$ and corresponding multipliers $\bar{\lambda} \geq 0$:

$$\bar{\lambda}^T [g(\bar{x}) + \nabla_I g(\bar{x})^T (x_I - \hat{x}_I)] \leq 0. \quad (\text{FCY}(\hat{x}))$$

In this way, a relaxed master problem similar to (MILP-OA) can be defined as:

$$\begin{aligned} z_{\text{GBD}(\text{KFS}, \text{KIS})} = \min \quad & \eta \\ \text{s.t. } & \eta \geq f(\bar{x}) + (\nabla_I f(\bar{x}) + \nabla_I g(\bar{x})\bar{\mu})^T (x_I - \bar{x}_I) \quad \forall \bar{x} \in \text{KFS}, \\ & \bar{\lambda}^T [g(\bar{x}) + \nabla_I g(\bar{x})^T (x_I - \bar{x}_I)] \leq 0 \quad \forall \bar{x} \in \text{KIS}, \\ & x \in X, \quad x_I \in \mathbb{Z}^I, \end{aligned} \quad (\text{RM-GBD})$$

where KFS is the set of solutions to feasible subproblems $(\text{NLP}(\hat{x}_I))$ and KIS is the set solutions to infeasible subproblems $(\text{NLPF}(\hat{x}_I))$. Convergence results for the GBD method are similar to those for OA.

THEOREM 3.2. [56] *If $X \neq \emptyset$, f and g are convex, and a constraint qualification holds for each $x^k \in K$, then $z_{\text{MINLP}} = z_{\text{GBD(KFS,KIS)}}$. The algorithm terminates in a finite number of steps.*

The inequalities used to create the master problem (RM-GBD) are aggregations of the inequalities used for (MILP-OA). As such, the lower bound obtained by solving a reduced version of (RM-GBD) (where only a subset of the constraints is considered) can be significantly weaker than for (MP(\mathcal{K})). This may explain why there is no available solver that uses solely the GBD method for solving convex MINLP. Abhishek, Leyffer and Linderoth [2] suggest to use the Benders cuts to aggregate inequalities in an LP/NLP-BB algorithm (see Section 3.5).

3.4. Extended Cutting Plane. Westerlund and Pettersson [111] proposed the Extended Cutting Plane (ECP) method for convex MINLPs, which is an extension of Kelley's cutting plane method [70] for solving convex NLPs. The ECP method was further extended to handle pseudoconvex function in the constraints [109] and in the objective [112] in the α -ECP method. Since this is beyond our definition of (MINLP), we give only here a description of the ECP method when all functions are convex. The reader is invited to refer to [110] for an up-to-date description of this enhanced method. The main feature of the ECP method is that it does not require the use of an NLP solver. The algorithm is based on the iterative solution of a reduced master problem (RM-ECP(\mathcal{K})). Linearizations of the most violated constraint at the optimal solution of (RM-ECP(\mathcal{K})) are added at every iteration. The MILP reduced master problem (RM-ECP(\mathcal{K})) is defined as:

$$\begin{aligned} z_{\text{ECP}(\mathcal{K})} = \min \quad & \eta \\ \text{s.t. } \eta \geq f(\bar{x}) + \nabla f(\bar{x})^T(x - \bar{x}) \quad & \bar{x} \in \mathcal{K} \quad (\text{RM-ECP}(\mathcal{K})) \\ g_j(\bar{x}) + \nabla g_j(\bar{x})^T(x - \bar{x}) \leq 0 \quad & \bar{x} \in \mathcal{K} \quad j \in J(\bar{x}) \\ x \in X, \quad x_I \in \mathbb{Z}^I \end{aligned}$$

where $J(\bar{x}) \stackrel{\text{def}}{=} \{j \in \arg \max_{j \in J} g_j(\bar{x})\}$ is the index set of most violated constraints for each solution $\bar{x} \in \mathcal{K}$, the set of solutions to (RM-ECP(\mathcal{K})). It is also possible to add linearizations of all violated constraints to (RM-ECP(\mathcal{K})). In that case, $J(\bar{x}) = \{j \mid g_j(\bar{x}) > 0\}$. Algorithm 3 gives the pseudocode for the ECP algorithm.

The optimal values $z_{\text{ECP}(\mathcal{K})}$ of (RM-ECP(\mathcal{K})) generate a non-decreasing sequence of lower bounds. Finite convergence of the algorithm is achieved when the maximum constraint violation is smaller than a specified tolerance ϵ . Theorem 3.3 states that the sequence of objective values obtained from the solutions to (RM-ECP(\mathcal{K})) converge to the optimal solution value.

THEOREM 3.3. [111] *If $X \neq \emptyset$ is compact and f and g are convex and continuously differentiable, then $z_{\text{ECP}(\mathcal{K})}$ converges to z_{MINLP} .*

The ECP method may require a large number of iterations, since the linearizations added at Step 3 are not coming from solutions to NLP subproblems. Convergence can often be accelerated by solving NLP subproblems ($NLP(\hat{x}_I)$) and adding the corresponding linearizations, as in the OA method. The Extended Cutting Plane algorithm is implemented in the α -ECP software [110].

Algorithm 3 The Extended Cutting Plane algorithm

0. **Initialize.**
Choose convergence tolerance ϵ . $\mathcal{K} \leftarrow \emptyset$.
 1. **Lower Bound**
Let $(\bar{\eta}^i, \bar{x}^i)$ be the optimal solution to $(RM-ECP(\mathcal{K}))$.
 2. **Terminate?**
Is $g_j(\bar{x}^i) < \epsilon \forall j \in J$ and $f(\bar{x}^i) - \bar{\eta}^i < \epsilon$? If so, \bar{x}^i is optimal with ϵ -feasibility.
 3. **Refine**
 $\mathcal{K} \leftarrow \mathcal{K} \cup \{\bar{x}^i\}$, $t \in \arg \max_j g_j(\bar{x}^i)$, and $J(\bar{x}^i) = \{t\}$
 $i \leftarrow i + 1$. Go to 1.
-

3.5. LP/NLP-Based Branch-and-Bound. The LP/NLP-Based Branch-and-Bound algorithm (LP/NLP-BB) was first proposed by Quezada and Grossmann [96]. The method is an extension of the OA method outlined in Section 3.2, but instead of solving a sequence of master problems ($MP(\mathcal{K})$), the master problem is dynamically updated in a single branch-and-bound tree that closely resembles the branch-and-cut method for MILP.

We denote by $LP(\mathcal{K}, \ell_I^i, u_I^i)$ the LP relaxation of $(MP(\mathcal{K}))$ obtained by dropping the integrality requirements and setting the lower and upper bounds on the x_I variables to ℓ_I and u_I respectively. The LP/NLP-BB method starts by solving the NLP relaxation $(NLPR(L_I, U_I))$, and sets up the reduced master problem $(MP(\mathcal{K}))$. A branch-and-bound enumeration is then started for $(MP(\mathcal{K}))$ using its LP relaxation. The branch-and-bound enumeration generates linear programs $LP(\mathcal{K}, \ell_I^i, u_I^i)$ at each node $N^i = (\ell_I^i, u_I^i)$ of the tree. Whenever an integer solution is found at a node, the standard branch and bound is interrupted and $(NLP(\hat{x}_I^i))$ (and $(NLPR(\hat{x}_I^i))$ if $NLP(\hat{x}_I^i)$ is infeasible) is solved by fixing integer variables to solution values at that node. The linearizations from the solution of $(NLP(\hat{x}_I^i))$ are then used to update the reduced master problem $(MP(\mathcal{K}))$. The branch-and-bound tree is then continued with the updated reduced master problem. The main advantage of LP/NLP-BB over OA is that the need of restarting the tree search is avoided and only a single tree is required. Algorithm 4 gives the pseudo-code for LP/NLP-BB.

Adding linearizations dynamically to the reduced master problem $(MP(\mathcal{K}))$ is a key feature of LP/NLP-BB. Note, however that the same

idea could potentially be applied to both the GBD and ECP methods. The LP/NLP-BB method commonly significantly reduces the total number of nodes to be enumerated when compared to the OA method. However, the trade-off is that the number of NLP subproblems might increase. As part of his Ph.D. thesis, Leyffer implemented the LP/NLP-BB method and reported substantial computational savings [76]. The LP/NLP-Based Branch-and-Bound algorithm is implemented in solvers Bonmin [24] and FilMINT [2].

Algorithm 4 The LP/NLP-Based Branch-and-Bound algorithm.

0. **Initialize.**
 $\mathcal{L} \leftarrow \{(L_I, U_I)\}$. $z_U \leftarrow +\infty$. $x^* \leftarrow \text{NONE}$.
 Let \bar{x} be the optimal solution of $(\text{NLPR}(l_I, u_I))$.
 $\mathcal{K} \leftarrow \{\bar{x}\}$.
 1. **Terminate?**
 Is $\mathcal{L} = \emptyset$? If so, the solution x^* is optimal.
 2. **Select.**
 Choose and delete a problem $N^i = (l_I^i, u_I^i)$ from \mathcal{L} .
 3. **Evaluate.**
 Solve $\text{LP}(\mathcal{K}, l_I^i, u_I^i)$. If the problem is infeasible, go to step 1, else let $z_{\text{MPR}(\mathcal{K}, l_I^i, u_I^i)}$ be its optimal objective function value and $(\hat{\eta}^i, \hat{x}^i)$ be its optimal solution.
 4. **Prune.**
 If $z_{\text{MPR}(\mathcal{K}, l_I^i, u_I^i)} \geq z_U$, go to step 1.
 5. **NLP Solve?**
 Is \hat{x}_I^i integer? If so, solve $(\text{NLP}(\hat{x}_I^i))$, otherwise go to step 8.
 Let \bar{x}^i be the optimal (or minimally infeasible) solution.
 6. **Upper bound?**
 Is \bar{x}^i feasible for (MINLP) and $f(\bar{x}^i) < z_U$? If so, $x^* \leftarrow \bar{x}^i$, $z_U \leftarrow f(\bar{x}^i)$.
 7. **Refine.**
 Let $\mathcal{K} \leftarrow \mathcal{K} \cup (\bar{x}^i)$. Go to step 3.
 8. **Divide.**
 Divide the feasible region of N^i into a number of smaller feasible subregions, creating nodes $N^{i_1}, N^{i_2}, \dots, N^{i_k}$. For each $j = 1, 2, \dots, k$, let $z_L^{ij} \leftarrow z_{\text{MPR}(\mathcal{K}, l_I^i, u_I^i)}$ and add the problem N^{ij} to \mathcal{L} . Go to step 1.
-

4. Implementation techniques for convex MINLP. Seasoned algorithmic developers know that proper engineering and implementation can have a large positive impact on the final performance of software. In this section, we present techniques that have proven useful in efficiently implementing the convex MINLP algorithms of Section 3.

The algorithms for solving MINLP we presented share a great deal in common with algorithms for solving MILP. NLP-BB is similar to a branch and bound for MILP, simply solving a different relaxation at each node. The LP/NLP-BB algorithm can be viewed as a branch-and-cut algorithm, similar to those employed to solve MILP, where the refining linearizations are an additional class of cuts used to approximate the feasible region. An MILP solver is used as a subproblem solver in the iterative algorithms (OA, GBD, ECP). In practice, all the methods spend most of their computing time doing variants of the branch-and-bound algorithm. As such, it stands to reason that advances in techniques for the implementation of branch and bound for MILP should be applicable and have a positive impact for solving MINLP. The reader is referred to the recent survey paper [84] for details about modern enhancements in MILP software.

First we discuss improvements to the **Refine** step of LP/NLP-BB, which may also be applicable to the GBD or ECP methods. We then proceed to the discussion of the **Select** and **Divide** steps which are important in any branch-and-bound implementation. The section contains an introduction to classes of cutting planes that may be useful for MINLP and reviews recent developments in heuristics for MINLP.

We note that in the case of iterative methods OA, GBD and ECP, some of these aspects are automatically taken care of by using a “black-box” MILP solver to solve $(MP(\mathcal{K}))$ as a component of the algorithm. In the case of NLP-BB and LP/NLP-BB, one has to more carefully take these aspects into account, in particular if one wants to be competitive in practice with methods employing MILP solvers as components.

4.1. Linearization generation. In the OA Algorithm 2, the ECP Algorithm 3, or the LP/NLP-BB Algorithm 4, a key step is to **Refine** the approximation of the nonlinear feasible region by adding linearizations of the objective and constraint functions (2.1) and (2.2). For convex MINLPs, linearizations may be generated at *any* point and still give a valid outer approximation of the feasible region, so for all of these algorithms, there is a mechanism for enhancing them by adding many linear inequalities. The situation is similar to the case of a branch-and-cut solver for MILP, where cutting planes such as Gomory cuts [59], mixed-integer-rounding cuts [85], and disjunctive (lift and project) cuts [9] can be added to approximate the convex hull of integer solutions, but care must be taken in a proper implementation to not overwhelm the software used for solving the relaxations by adding too many cuts. Thus, key to an effective refinement strategy in many algorithms for convex MINLP is a policy for deciding when inequalities should be added and removed from the master problem and at which points the linearizations should be taken.

Cut Addition: In the branch-and-cut algorithm for solving MILP, there is a fundamental implementation choice that must be made when confronted with an infeasible (fractional) solution: should the solution be

eliminated by cutting or branching? Based on standard ideas employed for answering this question in the context of MILP, we offer three rules-of-thumb that are likely to be effective in the context of linearization-based algorithms for solving MINLP. First, linearizations should be generated early in the procedure, especially at the very top of the branch-and-bound tree. Second, the incremental effectiveness of adding additional linearizations should be measured in terms of the improvement in the lower bound obtained. When the rate of lower bound change becomes too low, the refinement process should be stopped and the feasible region divided instead. Finally, care must be taken to not overwhelm the solver used for the relaxations of the master problem with too many linearizations.

Cut Removal: One simple strategy for limiting the number of linear inequalities in the continuous relaxation of the master problem ($\text{MP}(\mathcal{K})$) is to only add inequalities that are violated by the current solution to the linear program. Another simple strategy for controlling the size of ($\text{MP}(\mathcal{K})$) is to remove inactive constraints from the formulation. One technique is to monitor the dual variable for the row associated with the linearization. If the value of the dual variable is zero, implying that removal of the inequality would not change the optimal solution value, for many consecutive solutions, then the linearization is a good candidate to be removed from the master problem. To avoid cycling, the removed cuts are usually stored in a pool. Whenever a cut of the pool is found to be violated by the current solution it is put back into the formulation.

Linearization Point Selection. A fundamental question in any linearization-based algorithm (like OA, ECP, or LP/NLP-BB) is *at which points* should the linearizations be taken. Each algorithm specifies a minimal set of points at which linearizations must be taken in order to ensure convergence to the optimal solution. However, the algorithm performance may be improved by additional linearizations. Abhishek, Leyffer, and Linderoth [2] offer three suggestions for choosing points about which to take linearizations.

The first method simply linearizes the functions f and g about the fractional point \hat{x} obtained as a solution to an LP relaxation of the master problem. This method does not require the solution of an additional (non-linear) subproblem, merely the evaluation of the gradients of objective and constraint functions at the (already specified) point. (The reader will note the similarity to the ECP method).

A second alternative is to obtain linearizations about a point that is feasible with respect to the nonlinear constraints. Specifically, given a (possibly fractional) solution \hat{x} , the nonlinear program ($\text{NLP}(\hat{x}_I)$) is solved to obtain the point about which to linearize. This method has the advantage of generating linearization about points that are closer to the feasible region than the previous method, at the expense of solving the nonlinear program ($\text{NLP}(\hat{x}_I)$).

In the third point-selection method, no variables are fixed (save those that are fixed by the nodal subproblem), and the NLP relaxation ($\text{NLPR}(l_I, u_I)$) is solved to obtain a point about which to generate linearizations. These linearizations are likely to improve the lower bound by the largest amount when added to the master problem since the bound obtained after adding the inequalities is equal to $z_{\text{NLPR}(l_i, u_i)}$, but it can be time-consuming to compute the linearizations.

These three classes of linearizations span the trade-off spectrum of time required to generate the linearization versus the quality/strength of the resulting linearization. There are obviously additional methodologies that may be employed, giving the algorithm developer significant freedom to engineer linearization-based methods.

4.2. Branching rules. We now turn to the discussion of how to split a subproblem in the **Divide** step of the algorithms. As explained in Section 2.1, we consider here only branching by dichotomy on the variables. Suppose that we are at node N^i of the branch-and-bound tree with current solution \hat{x}^i . The goal is to select an integer-constrained variable $x_j \in I$ that is not currently integer feasible ($\hat{x}_j^i \notin \mathbb{Z}$) to create two subproblems by imposing the constraint $x_j \leq \lfloor \hat{x}_j^i \rfloor$ (*branching down*) and $x_j \geq \lceil \hat{x}_j^i \rceil$ (*branching up*) respectively. Ideally, one would want to select the variable that leads to the smallest enumeration tree. This of course cannot be performed exactly, since the variable which leads to the smallest subtree cannot be known *a priori*.

A common heuristic reasoning to choose the branching variable is to try to estimate how much one can improve the lower bound by branching on each variable. Because a node of the branch-and-bound tree is fathomed whenever the lower bound for the node is above the current upper bound, one should want to increase the lower bound as much as possible. Suppose that, for each variable x_j , we have estimates D_{j-}^i and D_{j+}^i on the increase in the lower bound value obtained by branching respectively down and up. A reasonable choice would be to select the variable for which both D_{j-}^i and D_{j+}^i are large. Usually, D_{j-}^i and D_{j+}^i are combined in order to compute a score for each variable and the variable of highest score is selected. A common formula for computing this score is:

$$\mu \min(D_{j-}^i, D_{j+}^i) + (1 - \mu) \max(D_{j-}^i, D_{j+}^i)$$

(where $\mu \in [0, 1]$ is a prescribed parameter typically larger than $\frac{1}{2}$).

As for the evaluation or estimation of D_{j-}^i and D_{j+}^i , two main methods have been proposed: pseudo-costs [17] and strong-branching [66, 6]. Next, we will present these two methods and how they can be combined.

4.2.1. Strong-branching. Strong-branching consists in computing the values D_{j-}^i and D_{j+}^i by performing the branching on variable x_j and solving the two associated sub-problems. For each variable x_j currently

fractional in \hat{x}_j^i , we solve the two subproblems N_{j-}^i and N_{j+}^i obtained by branching down and up, respectively, on variable j . Because N_{j-}^i and/or N_{j+}^i may be proven infeasible, depending on their status, different decision may be taken.

- If both sub-problems are infeasible: the node N^i is infeasible and is fathomed.
- If one of the subproblems is infeasible: the bound on variable x_j can be strengthened. Usually after the bound is modified, the node is reprocessed from the beginning (going back to the **Evaluate** step).
- If both subproblems are feasible, their values are used to compute D_{j-}^i and D_{j+}^i .

Strong-branching can very significantly reduce the number of nodes in a branch-and-bound tree, but is often slow overall due to the added computing cost of solving two subproblems for each fractional variable. To reduce the computational cost of strong-branching, it is often efficient to solve the subproblems only approximately. If the relaxation at hand is an LP (for instance in LP/NLP-BB) it can be done by limiting the number of dual simplex iterations when solving the subproblems. If the relaxation at hand is an NLP, it can be done by solving an approximation of the problem to solve. Two possible relaxations that have been recently suggested [23, 106, 80] are the LP relaxation obtained by constructing an Outer Approximation or the Quadratic Programming approximation given by the last Quadratic Programming sub-problem in a Sequential Quadratic Programming (SQP) solver for nonlinear programming (for background on SQP solvers see [94]).

4.2.2. Pseudo-costs. The pseudo-costs method consists in keeping the history of the effect of branching on each variable and utilizing this historical information to select good branching variables. For each variable x_j , we keep track of the number of times the variable has been branched on (τ_j) and the total per-unit degradation of the objective value by branching down and up, respectively, P_{j-} and P_{j+} . Each time variable j is branched on, P_{j-} and P_{j+} are updated by taking into account the change of bound at that node:

$$P_{j-} = \frac{z_L^{j-} - z_L^i}{f_j^i} + P_{j-}, \text{ and } P_{j+} = \frac{z_L^{j+} - z_L^i}{1 - f_j^i} + P_{j+},$$

where x_j is the branching variable, N_-^i and N_+^i denote the nodes from the down and up branch, z_L^i (resp. z_L^{j-} and z_L^{j+}) denote the lower bounds computed at node N^i (resp. N_-^i and N_+^i), and $f_j^i = \hat{x}_j^i - \lfloor \hat{x}_j^i \rfloor$ denotes the fractional part of \hat{x}_j^i . Whenever a branching decision has to be made, estimates of D_{j-}^i , D_{j+}^i are computed by multiplying the average of observed degradations with the current fractionality:

$$D_{j-}^i = f_j^i \frac{P_{j-}}{\tau_j}, \text{ and } D_{j+}^i = (1 - f_j^i) \frac{P_{j+}}{\tau_j}.$$

Note that contrary to strong-branching, pseudo-costs require very little computation since the two values P_{j-}^i and P_{j+}^i are only updated once the values z_L^{i-} and z_L^{i+} have been computed (by the normal process of branch and bound). Thus pseudo-costs have a negligible computational cost. Furthermore, statistical experiments have shown that pseudo-costs often provide reasonable estimates of the objective degradations caused by branching [83] when solving MILPs.

Two difficulties arise with pseudo-costs. The first one, is how to update the historical data when a node is infeasible. This matter is not settled. Typically, the pseudo-costs update is simply ignored if a node is infeasible.

The second question is how the estimates should be initialized. For this, it seems that the agreed upon state-of-the art is to combine pseudo-costs with strong-branching, a method that may address each of the two methods' drawbacks— strong-branching is too slow to be performed at every node of the tree, and pseudo-costs need to be initialized. The idea is to use strong-branching at the beginning of the tree search, and once all pseudo-costs have been initialized, to revert to using pseudo-costs. Several variants of this scheme have been proposed. A popular one is reliability branching [4]. This rule depends on a reliability parameter κ (usually a natural number between 1 and 8), pseudo-costs are trusted for a particular variable only after strong-branching has been performed κ times on this variable.

Finally, we note that while we have restricted ourselves in this discussion to dichotomy branching, one can branch in many different ways. Most state-of-the-art solvers allow branching on SOS constraints [14]. More generally, one could branch on *split disjunctions* of the form $(\pi^T x_I \leq \pi_0) \vee (\pi^T x_I \geq \pi_0 + 1)$ (where $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$). Although promising results have been obtained in the context of MILP [69, 37], as far as we know, these methods have not been used yet in the context of MINLPs. Finally, methods have been proposed to branch efficiently in the presence of symmetries [86, 95]. Again, although they would certainly be useful, these methods have not yet been adapted into software for solving MINLPs, though some preliminary work is being done in this direction [81].

4.3. Node selection rules. The other important strategic decision left unspecified in Algorithms 1 and 4 is which node to choose in the **Select** step. Here two goals needs to be considered: decreasing the global upper bound z^U by finding good feasible solutions, and proving the optimality of the current incumbent x^* by increasing the lower bound as fast as possible. Two classical node selection strategies are *depth-first-search* and *best-first* (or *best-bound*). As its name suggest, depth first search selects at each iteration the deepest node of the enumeration tree (or the last node put in \mathcal{L}). Best-first follows an opposite strategy of picking the open node N^i with the smallest z_L^i (the best lower bound).

Both these strategies have their inherent strengths and weaknesses. Depth-first has the advantage of keeping the size of the list of open-nodes as small as possible. Furthermore, the changes from one subproblem to the next are minimal, which can be very advantageous for subproblem solvers that can effectively exploit “warm-start” information. Also, depth-first search is usually able to find feasible solutions early in the tree search. On the other hand, depth-first can exhibit extremely poor performance if no good upper bound is known or found: it may explore many nodes with lower bound higher than the actual optimal solution. Best-bound has the opposite strengths and weaknesses. Its strength is that, for a fixed branching, it minimizes the number of nodes explored (because all nodes explored by it would be explored independently of the upper bound). Its weaknesses are that it may require significant memory to store the list \mathcal{L} of active nodes, and that it usually does not find integer feasible solutions before the end of the search. This last property may not be a shortcoming if the goal is to prove optimality but, as many applications are too large to be solved to optimality, it is particularly undesirable that a solver based only on best-first aborts after several hours of computing time without producing one feasible solution.

It should seem natural that good strategies are trying to combine both best-first and depth first. Two main approaches are *two-phase methods* [54, 13, 44, 83] and *diving methods* [83, 22].

Two-phase methods start by doing depth-first to find one (or a small number of) feasible solution. The algorithm then switches to best-first in order to try to prove optimality (if the tree grows very large, the method may switch back to depth-first to try to keep the size of the list of active nodes under control).

Diving methods are also two-phase methods in a sense. The first phase called *diving* does depth-first search until a leaf of the tree (either an integer feasible or an infeasible one) is found. When a leaf is found, the next node is selected by *backtracking* in the tree for example to the node with best lower bound, and another diving is performed from that node. The search continues by iterating diving and backtracking.

Many variants of these two methods have been proposed in context of solving MILP. Sometimes, they are combined with estimations of the quality of integer feasible solutions that may be found in a subtree computed using pseudo-costs (see for example [83]). Computationally, it is not clear which of these variants performs better. A variant of diving called probed diving that performs reasonably well was described by Bixby and Rothberg [22]. Instead of conducting a pure depth-first search in the diving phase, the probed diving method explores both children of the last node, continuing the dive from the best one of the two (in terms of bounds).

4.4. Cutting planes. Adding inequalities to the formulation so that its relaxation will more closely approximate the convex hull of integer fea-

sible solutions was a major reason for the vast improvement in MILP solution technology [22]. To our knowledge, very few, if any MINLP solvers add inequalities that are specific to the nonlinear structure of the problem. Nevertheless, a number of cutting plane techniques that could be implemented have been developed in the literature. Here we outline a few of these techniques. Most of them have been adapted from known methods in the MILP case. We refer the reader to [36] for a recent survey on cutting planes for MILP.

4.4.1. Gomory cuts. The earliest cutting planes for Mixed Integer Linear Programs were Gomory Cuts [58, 59]. For simplicity of exposition, we assume a pure Integer Linear Program (ILP): $I = \{1, \dots, n\}$, with linear constraints given in matrix form as $Ax \leq b$ and $x \geq 0$. The idea underlying the inequalities is to choose a set of non-negative multipliers $u \in \mathbb{R}_+^m$ and form the surrogate constraint $u^T Ax \leq u^T b$. Since $x \geq 0$, the inequality $\sum_{j \in N} \lfloor u^T a_j \rfloor x_j \leq u^T b$ is valid, and since $\lfloor u^T a_j \rfloor x_j$ is an integer, the right-hand side may also be rounded down to form the *Gomory cut* $\sum_{j \in N} \lfloor u^T a_j \rfloor x_j \leq \lfloor u^T b \rfloor$. This simple procedure suffices to generate *all* valid inequalities for an ILP [35]. Gomory cuts can be generalized to *Mixed Integer Gomory* (MIG) cuts which are valid for MILPs. After a period of not being used in practice to solve MILPs, Gomory cuts made a resurgence following the work of Balas *et al.* [10], which demonstrated that when used in combination with branch and bound, MIG cuts were quite effective in practice.

For MINLP, Cezik and Iyengar [34] demonstrate that if the nonlinear constraint set $g_j(x) \leq 0 \forall j \in J$ can be described using conic constraints $Tx \succeq_{\mathcal{K}} b$, then the Gomory procedure is still applicable. Here \mathcal{K} , is a homogeneous, self-dual, proper, convex cone, and the notation $x \succeq_{\mathcal{K}} y$ denotes that $(x - y) \in \mathcal{K}$. Each cone \mathcal{K} has a dual cone \mathcal{K}^* with the property that $\mathcal{K}^* \stackrel{\text{def}}{=} \{u \mid u^T z \geq 0 \forall z \in \mathcal{K}\}$. The extension of the Gomory procedure to the case of conic integer programming is clear from the following equivalence:

$$Ax \succeq_{\mathcal{K}} b \Leftrightarrow u^T Ax \geq u^T b \quad \forall u \succeq_{\mathcal{K}^*} 0.$$

Specifically, elements from the dual cone $u \in \mathcal{K}^*$ can be used to perform the aggregation, and the regular Gomory procedure applied. To the authors' knowledge, no current MINLP software employs conic Gomory cuts. However, most solvers generate Gomory cuts from the existing linear inequalities in the model. Further, as pointed out by Akrotirianakis, Maros, and Rustem [5], Gomory cuts may be generated from the linearizations (2.1) and (2.2) used in the OA, ECP, or LP/NLP-BB methods. Most linearization-based software will by default generate Gomory cuts on these linearizations.

4.4.2. Mixed integer rounding. Consider the simple two-variable set $X = \{(x_1, x_2) \in \mathbb{Z} \times \mathbb{R}_+ \mid x_1 \leq b + x_2\}$. It is easy to see that the

mixed integer rounding inequality $x_1 \leq \lfloor b \rfloor + \frac{1}{1-f}x_2$, where $f = b - \lfloor b \rfloor$ represents the fractional part of b , is a valid inequality for X . Studying the convex hull of this simple set and some related counterparts have generated a rich classes of inequalities that may significantly improve the ability to solve MILPs [85]. Key to generating useful inequalities for computation is to combine rows of the problem in a clever manner and to use variable substitution techniques.

Atamtürk and Narayan [7] have extended the concept of mixed integer rounding to the case of Mixed Integer Second-Order Cone Programming (MISOCP). For the conic mixed integer set

$$T = \left\{ (x_1, x_2, x_3) \in \mathbb{Z} \times \mathbb{R}^2 \mid \sqrt{(x_1 - b)^2 + x_2^2} \leq x_3 \right\}$$

the following *simple conic mixed integer rounding* inequality

$$\sqrt{[(1 - 2f)(x_1 - \lfloor b \rfloor) + f]^2 + x_2^2} \leq x_3$$

helps to describe the convex hull of T . They go on to show that employing these inequalities in a cut-and-branch procedure for solving MISOCPs is significantly beneficial. To the authors' knowledge, no available software employs this technology, so this may be a fruitful line of computational research.

4.4.3. Disjunctive inequalities. Stubbs and Mehrotra [101], building on the earlier seminal work of Balas [8] on disjunctive programming and its application to MILP (via lift and project cuts) of Balas, Ceria and Cornuéjols [9], derive a lift and project cutting plane framework for convex (0-1) MINLPs. Consider the feasible region of the continuous relaxation of (MINLP-1) $R = \{(x, \eta) \mid f(x) \leq \eta, g_j(x) \leq 0 \forall j \in J, x \in X\}$. The procedure begins by choosing a (branching) dichotomy $x_i = 0 \vee x_i = 1$ for some $i \in I$. The convex hull of the union of the two (convex) sets $R_i^- \stackrel{\text{def}}{=} \{(x, \eta) \in R \mid x_i = 0\}, R_i^+ = \{(x, \eta) \in R \mid x_i = 1\}$ can be represented in a space of dimension $3n + 5$ as

$$\mathcal{M}_i(R) = \left\{ \begin{array}{l} (x, \eta, x^-, \eta^-, \\ x^+, \eta^+, \lambda^-, \lambda^+) \end{array} \mid \begin{array}{l} x = \lambda^- x^- + \lambda^+ x^+, \\ \eta = \lambda^- \eta^- + \lambda^+ \eta^+, \\ \lambda^- + \lambda^+ = 1, \lambda^- \geq 0, \lambda^+ \geq 0 \\ (x^-, \eta^-) \in R_i^-, (x^+, \eta^+) \in R_i^+ \end{array} \right\}.$$

One possible complication with the convex hull description $\mathcal{M}_i(R)$ is caused by the nonlinear, nonconvex relationships $x = \lambda^- x^- + \lambda^+ x^+$ and $\eta = \lambda^- \eta^- + \lambda^+ \eta^+$. However, this description can be transformed to an equivalent description $\tilde{\mathcal{M}}_i(R)$ with only convex functional relationships between variables using the perspective function [101, 65].

Given some solution $(\bar{x}, \bar{\eta}) \notin \text{conv}(R_i^- \cup R_i^+)$, the lift and project procedure operates by solving a convex separation problem

$$\min_{(x, \eta, \bar{x}^-, \bar{\eta}^-, \bar{x}^+, \bar{\eta}^+, \lambda^-, \lambda^+) \in \tilde{\mathcal{M}}_i(R)} d(x, \eta) \quad (4.1)$$

(where $d(x, \eta)$ is the distance to the point $(\bar{x}, \bar{\eta})$ in any norm). The lift-and-project inequality

$$\xi_x^T(x - \bar{x}) + \xi_\eta^T(\eta - \bar{\eta}) \geq 0 \quad (4.2)$$

separates $(\bar{x}, \bar{\eta})$ from $\text{conv}(R_i^- \cup R_i^+)$, where ξ is a subgradient of $d(x, \eta)$ at the optimal solution of (4.1).

An implementation and evaluation of some of these ideas in the context of MISOCP has been done by Drewes [41]. Cezik and Iyengar [34] had also stated that the application of disjunctive cuts to conic-IP should be possible.

A limitation of disjunctive inequalities is that in order to generate a valid cut, one must solve an auxiliary (separation) problem that is three times the size of the original relaxation. In the case of MILP, clever intuition of Balas and Perregaard [11] allow to solve this separation problem while staying in the original space. No such extension is known in the case of MINLP. Zhu and Kuno [114] have suggested to replace the true nonlinear convex hull by a linear approximation taken about the solution to a linearized master problem like $\text{MP}(\mathcal{K})$.

Kılınç *et al.* [71] have recently made the observation that a weaker form of the lift and project inequality (4.2) can be obtained from branching dichotomy information. Specifically, given values $\hat{\eta}_i^- = \min\{\eta | (x, \eta) \in R_i^-\}$ and $\hat{\eta}_i^+ = \min\{\eta | (x, \eta) \in R_i^+\}$, the *strong-branching cut*

$$\eta \geq \hat{\eta}_i^- + (\hat{\eta}_i^+ - \hat{\eta}_i^-)x_i$$

is valid for MINLP, and is a special case of (4.2). Note that if strong-branching is used to determine the branching variable, then the values $\hat{\eta}_i^-$ and $\hat{\eta}_i^+$ are produced as a byproduct.

4.5. Heuristics. Here we discuss heuristic methods that are aimed at finding integer feasible solutions to MINLP with no guarantee of optimality or success. Heuristics are usually fast algorithms. In a branch-and-bound algorithm they are typically run right after the **Evaluate** step. Depending on the actual running time of the heuristic, it may be called at every node, every n^{th} node, or only at the root node. In linearization-based methods like OA, GBD, ECP, or LP/NLP-BB, heuristics may be run in the **Upper Bound** and **Refine** step, especially in the case when $\text{NLP}(\hat{x}_I)$ is infeasible. Heuristics are very important because by improving the upper bound z_U , they help in the **Prune** step of the branch-and-bound algorithm or in the convergence criterion of the other algorithms. From a practical point of

view, heuristics are extremely important when the algorithm cannot be carried out to completion, so that a feasible solution may be returned to the user.

Many heuristic methods have been devised for MILP, we refer the reader to [19] for a recent and fairly complete review. For convex MINLP, two heuristic principles that have been used are *diving heuristics* and the *feasibility pump*.

We note that several other heuristic principles could be used such as RINS [39] or Local Branching [49] but as far as we know, these have not been applied yet to (convex) MINLPs and we will not cover them here.

4.5.1. Diving heuristics. Diving heuristics are very related to the diving strategies for node selection presented in Section 4.3. The basic principle is to simulate a dive from the current node to a leaf of the tree by fixing variables (either one at a time or several variables at a time).

The most basic scheme is, after the NLP relaxation has been solved, to fix the variable which is the least integer infeasible in the current solution to the closest integer and resolve. This process is iterated until either the current solution is integer feasible or the NLP relaxation becomes infeasible. Many variants of this scheme have been proposed for MILP (see [19] for a good review). These differ mainly in the the number of variables fixed, the way to select variables to fix, and in the possibility of doing a certain amount of *backtracking* (unfixing previously fixed variables). The main difficulty when one tries to adapt these scheme to MINLP is that instead of having to resolve an LP with a modified bound at each iteration (an operation which is typically done extremely efficiently by state-of-the-art LP solvers) one has to solve an NLP (where warm-starting methods are usually much less efficient).

Bonami and Gonçalves [26] have adapted the basic scheme to MINLPs in two different manners. First in a straightforward way, but trying to limit the number of NLPs solved by fixing many variables at each iteration and backtracking if the fixings induce infeasibility. The second adaptation tries to reduce the problem to a MILP by fixing all the variables that appear in a nonlinear term in the objective or the constraints. This MILP problem may be given to a MILP solver in order to find a feasible solution. A similar MINLP heuristic idea of fixing variables to create MILPs can be found in the work [20].

4.5.2. Feasibility pumps. The feasibility pump is another heuristic principle for quickly finding feasible solutions. It was initially proposed by Fischetti, Glover and Lodi [48] for MILP, and can be extended to convex MINLP in several manners.

First we present the feasibility pump in its most trivial extension to MINLP. The basic principle of the Feasibility Pump consists of generating a sequence of points $\bar{x}^0, \dots, \bar{x}^k$ that satisfy the constraints of the continuous relaxation $\text{NLPR}(L_I, U_I)$. Associated with the sequence $\bar{x}^0, \dots, \bar{x}^k$

of integer infeasible points is a sequence $\hat{x}^1, \dots, \hat{x}^{k+1}$ of points that are integer feasible but do not necessarily satisfy the other constraints of the problem. Specifically, \bar{x}^0 is the optimal solution of $\text{NLPR}(L_I, U_I)$. Each \hat{x}^{i+1} is obtained by rounding \bar{x}_j^i to the nearest integer for each $j \in I$ and keeping the other components equal to \bar{x}_j^i . The sequence \bar{x}^i is generated by solving a nonlinear program whose objective function is to minimize the distance of x to \hat{x}^i on the integer variables according to the ℓ_1 -norm:

$$\begin{aligned} z_{\text{FP-NLP}(\hat{x}_I)} = & \text{minimize} \quad \sum_{k \in I} |x_k - \hat{x}_k^i| \\ \text{subject to} \quad & g_j(x) \leq 0 \quad \forall j \in J, \\ & x \in X; L_I \leq \hat{x}_I \leq U_I. \end{aligned} \quad (\text{FP-NLP}(\hat{x}_I))$$

The two sequences have the property that at each iteration the distance between \bar{x}^i and \hat{x}^{i+1} is non-increasing. The procedure stops whenever an integer feasible solution is found (or $\hat{x}^k = \bar{x}^k$). This basic procedure may cycle or stall without finding an integer feasible solution and randomization has been suggested to restart the procedure [48]. Several variants of this basic procedure have been proposed in the context of MILP [18, 3, 50]. The authors of [1, 26] have shown that the basic principle of the Feasibility Pump can also find good solutions in short computing time also in the context of MINLP.

Another variant of the Feasibility Pump for convex MINLPs was proposed by Bonami *et al.* [25]. Like in the basic FP scheme two sequences are constructed with the same properties: $\bar{x}^0, \dots, \bar{x}^k$ are points in X that satisfy $g(\bar{x}^i) \leq 0$ but not $\bar{x}_I^i \in \mathbb{Z}^{|I|}$ and $\hat{x}^1, \dots, \hat{x}^{k+1}$ are points that do not necessarily satisfy $g(\hat{x}^i) \leq 0$ but satisfy $\hat{x}_I^i \in \mathbb{Z}^{|I|}$. The sequence \bar{x}^i is generated in the same way as before but the sequence \hat{x}^i is now generated by solving MILPs. The MILP to solve for finding \hat{x}^{i+1} is constructed by building an outer approximation of the constraints of the problem with linearizations taken in all the points of the sequence $\bar{x}^0, \dots, \bar{x}^i$. Then, \hat{x}^{i+1} is found as the point in the current outer approximation of the constraints that is closest to \bar{x}^i in ℓ_1 -norm in the space of integer constrained variables:

$$\begin{aligned} z_{\text{FP-M}^i} = & \text{minimize} \sum_{i \in I} |x_j - \bar{x}_j^i| \\ \text{s.t. } & g(\bar{x}^l) + \nabla g(\bar{x}^l)^T(x - \bar{x}^l) \leq 0 \quad l = 1, \dots, i, \\ & x \in X, \quad x_I \in \mathbb{Z}^I. \end{aligned} \quad (\text{M-FP}^i)$$

Unlike the procedure of Fischetti, Glover and Lodi, the *Feasibility Pump for MINLP* cannot cycle and it is therefore an exact algorithm: either it finds a feasible solution or it proves that none exists. This variant of the FP principle can also be seen as a variant of the Outer Approximation decomposition scheme presented in Section 3.2. In [25], it was also proposed

to iterate the FP scheme by integrating the linearization of the objective function in the constraint system of (M-FP^{*i*}) turning the feasibility pump into an exact iterative algorithm which finds solutions of increasingly better cost until eventually proving optimality. Abhishek *et al.* [1] have also proposed to integrate this Feasibility Pump into a single tree search (in the same way as Outer Approximation decomposition can be integrated in a single tree search when doing the LP/NLP-BB).

5. Software. There are many modern software packages implementing the algorithms of Section 3 that employ the modern enhancements described in Section 5. In this section, we describe the features of six different packages. The focus is on solvers for *general* convex MINLPs, not only special cases such as MIQP, MIQCP, or MISOCP. All of these packages may be freely used via a web interface on NEOS (<http://www-neos.mcs.anl.gov>).

5.1. α -ECP. α -ECP [110] is a solver based on the ECP method described in Section 3.4. Problems to be solved may be specified in a text-based format, as user-supplied subroutines, or via the GAMS algebraic modeling language. The software is designed to solve convex MINLP, but problems with a pseudo-convex objective function and pseudo-convex constraints can also be solved to global optimality with α -ECP. A significant feature of the software is that *no* nonlinear subproblems are required to be solved. (Though recent versions of the code have included an option to occasionally solve NLP subproblems, which may improve performance, especially on pseudo-convex instances.) Recent versions of the software also include enhancements so that each MILP subproblem need not be solved to global optimality. α -ECP requires a (commercial) MILP software to solve the reduced master problem (RM-ECP(\mathcal{K})), and CPLEX, XPRESS-MP, or Mosek may be used for this purpose.

In the computational experiment of Section 6, α -ECP (v1.75.03) is used with CPLEX (v12.1) as MILP solver, CONOPT (v3.24T) as NLP solver and α -ECP is run via GAMS. Since all instances are convex, setting the *ECPstrategy* option to 1 instructed α -ECP to not perform algorithmic steps relating to the solution of pseudo-convex instances.

5.2. Bonmin. Bonmin is an open-source MINLP solver and framework with implementations of algorithms NLP-BB, OA, and two different LP/NLP-BB algorithms with different default parameters. Source code and binaries of Bonmin are available from COIN-OR (<http://www.coin-or.org>). Bonmin may be called as a solver from both the AMPL and GAMS modeling languages.

Bonmin interacts with the COIN-OR software Cbc to manage the branch-and-bound trees of its various algorithms. To solve NLP subproblems, Bonmin may be instrumented to use either Ipopt [107] or FilterSQP [52]. Bonmin uses the COIN-OR software Clp to solve linear programs,

and may use Cbc or Cplex to solve MILP subproblems arising in its various algorithms.

The Bonmin NLP-BB algorithm features a range of different heuristics, advanced branching techniques such as strong-branching or pseudo-costs branching, and five different choices for node selection strategy. The Bonmin LP/NLP-BB methods use row management, cutting planes, and branching strategies from Cbc. A distinguishing feature of Bonmin is that one may instruct Bonmin to use a (time-limited) OA or feasibility pump heuristic at the beginning of the optimization.

In the computational experiments, Bonmin (v1.1) is used with Cbc (v2.3) as the MILP solver, Ipopt (v2.7) as NLP solver, and Clp (v1.10) is used as LP solver. For Bonmin, the algorithms, NLP-BB (denoted as B-BB) and LP/NLP-BB (denoted as B-Hyb) are tested. The default search strategies of dynamic node selection (mixture of depth-first-search and best-bound) and strong-branching were employed.

5.3. DICOPT. DICOPT is a software implementation of the OA method described in Section 3.2. DICOPT may be used as a solver from the GAMS modeling language. Although OA has been designed to solve convex MINLP, DICOPT may often be used successfully as a heuristic approach for nonconvex MINLP, as it contains features such as equality relaxation [72] and augmented penalty methods [105] for dealing with nonconvexities. DICOPT requires solvers for both NLP subproblems and MILP subproblems, and it uses available software as a “black-box” in each case. For NLP subproblems, possible NLP solvers include CONOPT [42], MINOS [89] and SNOPT [57]. For MILP subproblems, possible MILP solvers include CPLEX [66] and XPRESS [47]. DICOPT contains a number of heuristic (inexact) stopping rules for the OA method that may be especially effective for nonconvex instances.

In our computational experiment, the DICOPT that comes with GAMS v23.2.1 is used with CONOPT (v3.24T) as the NLP solver and Cplex (v12.1) as the MILP solver. In order to ensure that instances are solved to provable optimality, the GAMS/DICOPT option `stop` was set to value 1.

5.4. FilMINT. FilMINT [2] is a non-commercial solver for convex MINLPs based on the LP/NLP-BB algorithm. FilMINT may be used through the AMPL language.

FilMINT uses MINTO [93] a branch-and-cut framework for MILP to solve the reduced master problem ($\text{MP}(\mathcal{K})$) and filterSQP [52] to solve nonlinear subproblems. FilMINT uses the COIN-OR LP solver Clp or CPLEX to solve linear programs.

FilMINT by default employs nearly all of MINTO’s enhanced MILP features, such as cutting planes, primal heuristics, row management, and enhanced branching and node selection rules. By default, pseudo-costs branching is used as branching strategy and best estimate is used as node

selection strategy. An NLP-based Feasibility Pump can be run at the beginning of the optimization as a heuristic procedure. The newest version of FilMINT has been augmented with the simple strong-branching disjunctive cuts described in Section 4.4.3.

In the computational experiments of Section 6, FilMINT v0.1 is used with Clp as LP solver. Two versions of FilmINT are tested—the default version and a version including the strong-branching cuts (Filmint-SBC).

5.5. MINLP_BB. MINLP_BB [77] is an implementation of the NLP-BB algorithm equipped with different node selection and variable selection rules. Instances can be specified to MINLP_BB through an AMPL interface.

MINLP_BB contains its own tree-manager implementation, and NLP subproblems are solved by FilterSQP [52]. Node selection strategies available in MINLP_BB include depth-first-search, depth-first-search with backtrack to best-bound, best-bound, and best-estimated solution. For branching strategies, MINLP_BB contains implementations of most fractional branching, strong-branching, approximate strong-branching using second-order information, pseudo-costs branching and reliability branching.

MINLP_BB is written in FORTRAN. Thus, there is no dynamic memory allocation, and the user must specify a maximum memory (stack) size at the beginning of algorithm to store the list of open nodes.

For the computational experiments with MINLP_BB, different levels of stack size were tried in an attempt to use the entire available memory for each instance. The default search strategies of depth-first-search with backtrack to best-bound and pseudo-costs branching were employed in MINLP_BB (v20090811).

5.6. SBB. SBB [30] is a NLP-based branch-and-bound solver that is available through the GAMS modeling language. The NLP subproblems can be solved by CONOPT [42], MINOS [89] and SNOPT [57]. Pseudo-costs branching is an option as a branching rule. As a node selection strategy, depth-first-search, best-bound, best-estimate or combination of these three can be employed. Communication of subproblems between the NLP solver and tree manager is done via files, so SBB may incur some extra overhead when compared to other solvers.

In our computational experiments, we use the version of SBB shipped with GAMS v23.2.1. CONOPT is used as NLP solver, and the SBB default branching variable and node selection strategies are used.

6. Computational study.

6.1. Problems. The test instances used in the computational experiments were gathered from the MacMINLP collection of test problems [79], the GAMS collection of MINLP problems [31], the collection on the website of IBM-CMU research group [99], and instances created by the authors. Characteristics of the instances are given in [Table 1](#), which lists whether

or not the instance has a nonlinear objective function, the total number of variables, the number of integer variables, the number of constraints, and how many of the constraints are nonlinear.

BatchS: The *BatchS* problems [97, 104] are multi-product batch plant design problems where the objective is to determine the volume of the equipment, the number of units to operate in parallel, and the locations of intermediate storage tanks.

CLay: The *CLay* problems [98] are constrained layout problems where non-overlapping rectangular units must be placed within the confines of certain designated areas such that the cost of connecting these units is minimized.

FLay: The *FLay* problems [98] are farmland layout problems where the objective is to determine the optimal length and width of a number of rectangular patches of land with fixed area, such that the perimeter of the set of patches is minimized.

fo-m-o: These are block layout design problems [33], where an orthogonal arrangement of rectangular departments within a given rectangular facility is required. A distance-based objective function is to be minimized, and the length and width of each department should satisfy given size and area requirements.

RSyn: The *RSyn* problems [98] concern retrofit planning, where one would like to redesign existing plants to increase throughput, reduce energy consumption, improve yields, and reduce waste generation. Given limited capital investments to make process improvements and cost estimations over a given time horizon, the problem is to identify the modifications that yield the highest income from product sales minus the cost of raw materials, energy, and process modifications.

SLay: The *SLay* problems [98] are safety layout problems where optimal placement of a set of units with fixed width and length is determined such that the Euclidean distance between their center point and a predefined “safety point” is minimized.

sssd: The *sssd* instances [45] are stochastic service system design problems. Servers are modeled as M/M/1 queues, and a set of customers must be assigned to the servers which can be operated at different service levels. The objective is to minimize assignment and operating costs.

Syn: The *Syn* instances [43, 102] are synthesis design problems dealing with the selection of optimal configuration and parameters for a processing system selected from a superstructure containing alternative processing units and interconnections.

trimloss: The *trimloss* (*tls*) problems [64] are cutting stock problems where one would like to determine how to cut out a set of product paper rolls from raw paper rolls such that the trim loss as well as the overall production is minimized.

uflquad: The *uflquad* problems [61] are (separable) quadratic uncapacitated facility location problems where a set of customer demands must be

TABLE 1
Test set statistics.

Problem	NonL Obj	Vars	Ints	Cons	NonL Cons
BatchS121208M	✓	407	203	1510	1
BatchS151208M	✓	446	203	1780	1
BatchS201210M	✓	559	251	2326	1
CLay0303H		100	21	114	36
CLay0304H		177	36	210	48
CLay0304M		57	36	58	48
CLay0305H		276	55	335	60
CLay0305M		86	55	95	60
FLay04H		235	24	278	4
FLay05H		383	40	460	5
FLay05M		63	40	60	5
FLay06M		87	60	87	6
fo7_2		115	42	197	14
fo7		115	42	197	14
fo8		147	56	257	16
m6		87	30	145	12
m7		115	42	197	14
o7_2		115	42	197	14
RSyn0805H		309	37	426	3
RSyn0805M02M		361	148	763	6
RSyn0805M03M		541	222	1275	9
RSyn0805M04M		721	296	1874	12
RSyn0810M02M		411	168	854	12
RSyn0810M03M		616	252	1434	18
RSyn0820M		216	84	357	14
RSyn0830M04H		2345	496	4156	80
RSyn0830M		251	94	405	20
RSyn0840M		281	104	456	28
SLay06H	✓	343	60	435	0
SLay07H	✓	477	84	609	0
SLay08H	✓	633	112	812	0
SLay09H	✓	811	144	1044	0
SLay09M	✓	235	144	324	0
SLay10M	✓	291	180	405	0
sssd-10-4-3		69	52	30	12
sssd-12-5-3		96	75	37	15
sssd-15-6-3		133	108	45	18
Syn15M04M		341	120	762	44
Syn20M03M		316	120	657	42
Syn20M04M		421	160	996	56
Syn30M02M		321	120	564	40
Syn40M03H		1147	240	1914	84
Syn40M		131	40	198	28
tls4		106	89	60	4
tls5		162	136	85	5
uflquad-20-150	✓	3021	20	3150	0
uflquad-30-100	✓	3031	30	3100	0
uflquad-40-80	✓	3241	40	3280	0

satisfied by open facilities. The objective is to minimize the sum of the fixed cost for operating facilities and the shipping cost which is proportional to the square of the quantity delivered to each customer.

All test problems are available in AMPL and GAMS formats and are available from the authors upon request. In our experiments, α -ECP, DICOPT, and SBB are tested through the GAMS interface, while Bonmin, FilMINT and MINLP_BB are tested through AMPL.

6.2. Computational results. The computational experiments have been run on a cluster of identical 64-bit Intel Xeon microprocessors clocked at 2.67 GHz, each with 3 GB RAM. All machines run the Red Hat Enterprise Linux Server 5.3 operating system. A three hour time limit is enforced. The computing times used in our comparisons are the wall-clock times (including system time). All runs were made on processors dedicated to the computation. Wall-clock times were used to accurately account for system overhead incurred by file I/O operations required by the SBB solver. For example, on the problem F_{Lay}05M, SBB reports a solution time of 0.0 seconds for 92241 nodes, but the wall-clock time spent is more than 17 minutes.

[Table 3](#) summarizes the performance of the solvers on the 48 problems of the test set. The table lists for each solver the number of times the optimal solution was found, the number of times the time limit was exceeded, the number of times the node limit exceeded, the number of times an error occurred (other than time limit or memory limit), the number of times the solver is fastest, and the arithmetic and geometric means of solution times in seconds. When reporting aggregated solution times, unsolved or failed instances are accounted for with the time limit of three hours. A performance profile [40] of solution time is given in [Figure 1](#). The detailed performance of each solver on each test instance is listed in [Table 4](#).

There are a number of interesting observations that can be made from this experiment. First, for the instances that they can solve, the solvers DICOPT and α -ECP tend to be very fast. Also, loosely speaking, for each class of instances, there seems to be one or two solvers whose performance dominates the others, and we have listed these in [Table 2](#).

TABLE 2
Subjective Rating of Best Solver on Specific Instance Families.

Instance Family	Best Solvers
<i>Batch</i>	DICOPT
<i>CLay, F_{Lay}, ssss</i>	FilMINT, MINLP_BB
<i>Fo, RSyn, Syn</i>	DICOPT, α -ECP
<i>SLay</i>	MINLP_BB
<i>ufquad</i>	Bonmin (B-BB)

TABLE 3
Solver statistics on the test set.

Solver	Opt.	Time Limit	Mem. Limit	Error	Fastest	Arith. Mean	Geom. Mean
α -ECP	37	9	0	2	4	2891.06	105.15
Bonmin-BB	35	5	8	0	4	4139.60	602.80
Bonmin-Hyb	32	0	15	1	1	3869.08	244.41
Dicopt	30	16	0	2	21	4282.77	90.79
Filmint	41	7	0	0	4	2588.79	343.47
Filmint-SBC	43	5	0	0	3	2230.11	274.61
MinlpBB	35	3	7	3	12	3605.45	310.09
Sbb	18	23	6	1	0	7097.49	2883.75

In general, the variation in solver performance on different instance families indicates that a “portfolio” approach to solving convex MINLPs is still required. Specifically, if the performance of a specific solver is not satisfactory, one should try other software on the instance as well.

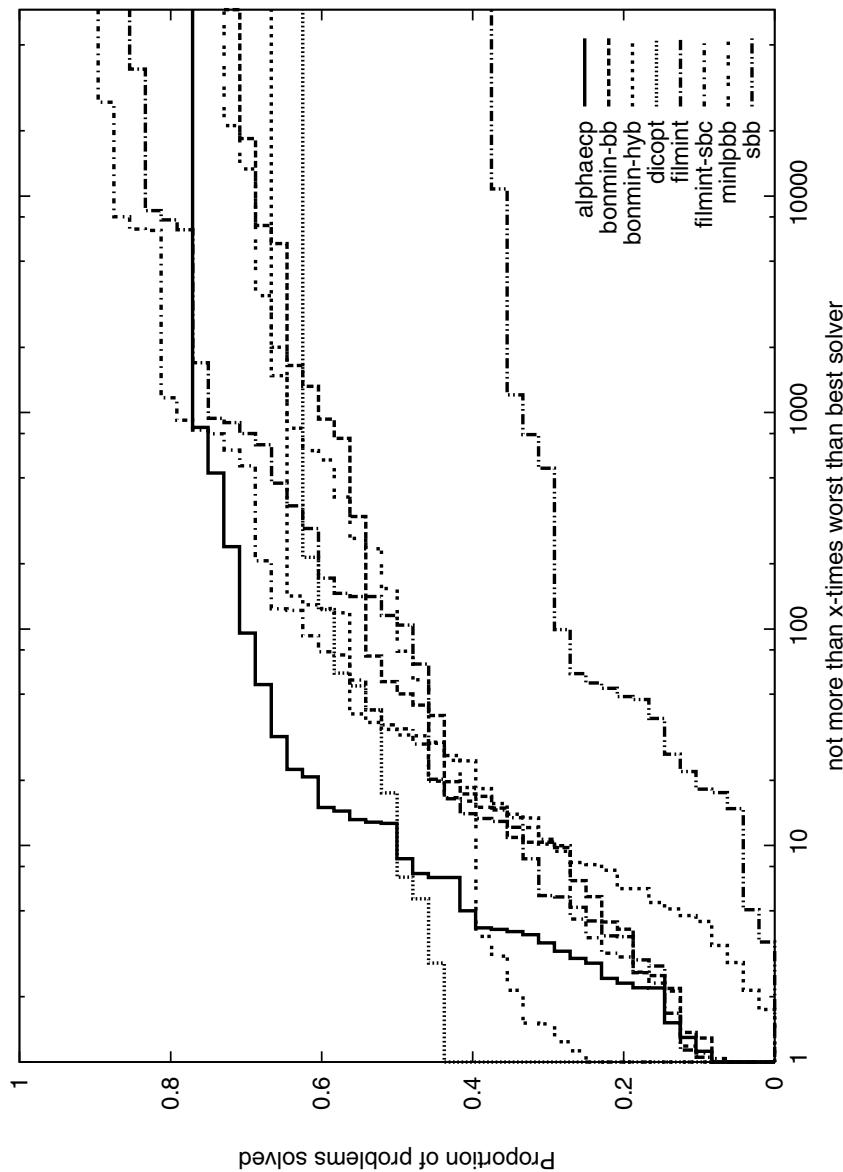
7. Conclusions. Convex Mixed-Integer Nonlinear Programs (MINLPs) can be used to model many decision problems involving both nonlinear and discrete components. Given their generality and flexibility, MINLPs have been proposed for many diverse and important scientific applications. Algorithms and software are evolving so that instances of these important models can often be solved in practice. The main advances are being made along two fronts. First, new theory is being developed. Second, theory and implementation techniques are being translated from the more-developed arena of Mixed Integer Linear Programming into MINLP. We hope this survey has provided readers the necessary background to delve deeper into this rapidly evolving field.

Acknowledgments. The authors would also like to thank Jon Lee and Sven Leyffer for inviting them to the stimulating IMA workshop on Mixed Integer Nonlinear Programming. The comments of two anonymous referees greatly improved the presentation.

TABLE 4

Comparison of running times (in seconds) for the solvers α -ECP(α **ECP**), Bonmin-BB(**B-BB**), Bonmin-LP/NLP-BB(**B-Hyb**), DICOPT, FilmINT(**Fil**), FilmINT with strong-branching cuts(**Fil-SBC**), MINLP-BB(**M-BB**) and SBB (bold face for best running time). If the solver could not provide the optimal solution, we state the reason with following letters: “t” states that the 3 hour time limit is hit, “m” states that the 3 GB memory limit is passed over and “f” states that the solver has failed to find optimal solution without hitting time limit or memory limit.

Problem	α ECP	B-BB	B-Hyb	Dicotp	Fil	Fil-SBC	M-BB	Sbb
BatchS121208M	384.8	47.8	43.9	6.9	31.1	14.7	128.7	690.1
BatchS151208M	297.4	139.2	71.7	9.3	124.5	42.8	1433.5	138.3
BatchS201210M	137.3	188.1	148.8	9.5	192.4	101.9	751.2	463.8
CLay0303H	7.0	21.1	13.0	33.2	2.0	1.4	0.5	14.0
CLay0304H	22.1	76.0	68.9	t	21.7	8.7	7.3	456.6
CLay0304M	16.2	54.1	50.4	t	5.0	5.0	6.2	192.4
CLay0305H	88.8	2605.5	125.2	1024.3	162.0	58.4	87.6	1285.6
CLay0305M	22.7	775.0	46.1	2211.0	10.3	32.9	31.9	582.6
FLay04H	106.0	48.1	42.2	452.3	8.3	8.6	12.5	41.9
FLay05H	t	2714.0	m	t	1301.2	1363.4	1237.8	4437.0
FLay05M	5522.3	596.8	m	t	698.4	775.4	57.5	1049.2
FLay06M	t	m	m	t	t	t	2933.6	t
fo7_2	16.5	m	104.5	t	271.9	200.0	964.1	t
fo7	98.9	t	285.2	4179.6	1280.1	1487.9	f	t
fo8	447.4	t	m	t	3882.7	7455.5	m	t
m6	0.9	79.4	31.1	0.2	89.0	29.8	7.1	2589.9
m7	4.6	3198.3	75.4	0.5	498.6	620.0	215.9	t
o7_2	728.4	m	m	t	3781.3	7283.8	m	t
RSyn0805H	0.6	4.0	0.1	0.1	0.5	3.4	2.0	4.3
RSyn0805M02M	9.8	2608.2	m	3.4	485.0	123.8	806.1	t
RSyn0805M03M	16.6	6684.6	10629.9	7.2	828.2	668.3	4791.9	t
RSyn0805M04M	10.5	10680.0	m	8.1	1179.2	983.6	4878.8	m
RSyn0810M02M	10.1	t	m	4.6	7782.2	3078.1	m	m
RSyn0810M03M	43.3	m	m	10.8	t	8969.5	m	m
RSyn0820M	1.4	5351.4	11.8	0.3	232.2	231.6	1005.1	t
RSyn0830M04H	19.9	320.4	30.5	5.6	78.7	193.7	f	f
RSyn0830M	2.2	m	7.4	0.5	375.7	301.1	1062.3	m
RSyn0840M	1.6	m	13.7	0.4	3426.2	2814.7	m	m
SLay06H	316.8	3.4	34.4	7.5	186.6	5.0	1.3	731.9
SLay07H	2936.0	7.6	46.6	39.8	385.0	81.5	5.6	t
SLay08H	8298.0	12.5	178.1	t	1015.0	156.0	9.7	t
SLay09H	t	25.6	344.0	3152.5	7461.5	1491.9	55.0	t
SLay09M	t	8.6	63.0	t	991.7	41.8	2.1	1660.7
SLay10M	t	108.4	353.7	t	t	3289.6	43.3	2043.7
sssd-10-4-3	2.4	16.8	31.2	2.8	3.7	3.0	1.0	t
sssd-12-5-3	f	56.1	m	f	9.6	13.3	5.7	t
sssd-15-6-3	f	163.4	m	f	36.9	1086.4	41.6	t
Syn15M04M	2.9	379.5	8.5	0.2	39.5	47.5	60.4	278.0
Syn20M03M	2.9	9441.4	16.2	0.1	1010.3	901.9	1735.0	t
Syn20M04M	3.9	m	22.6	0.2	7340.9	5160.4	m	t
Syn30M02M	3.0	t	13.6	0.4	2935.5	3373.8	8896.1	t
Syn40M03H	8.3	18.9	3.9	1.1	6.5	87.9	f	19.7
Syn40M	1.8	877.7	0.6	0.1	108.1	110.7	101.4	t
tls4	377.7	t	f	t	383.0	336.7	1281.8	t
tls5	t	m	m	t	t	t	m	m
uflquad-20-150	t	422.2	m	t	t	t	t	t
uflquad-30-100	t	614.5	m	t	t	t	t	t
uflquad-40-80	t	9952.3	m	t	t	t	t	t

FIG. 1. *Performance Profile Comparing Convex MINLP Solvers.*

REFERENCES

- [1] K. ABHISHEK, S. LEYFFER, AND J.T. LINDEROTH, *Feasibility pump heuristics for Mixed Integer Nonlinear Programs*. Unpublished working paper, 2008.
- [2] ———, *FilmINT: An outer-approximation-based solver for convex Mixed-Integer Nonlinear Programs*, INFORMS Journal on Computing, **22**, No. 4 (2010), pp. 555–567.
- [3] T. ACHTERBERG AND T. BERTHOLD, *Improving the feasibility pump*, Technical Report ZIB-Report 05-42, Zuse Institute Berlin, September 2005.
- [4] T. ACHTERBERG, T. KOCH, AND A. MARTIN, *Branching rules revisited*, Operations Research Letters, **33** (2004), pp. 42–54.
- [5] I. AKROTIRIANAKIS, I. MAROS, AND B. RUSTEM, *An outer approximation based branch-and-cut algorithm for convex 0-1 MINLP problems*, Optimization Methods and Software, **16** (2001), pp. 21–47.
- [6] D. APPLEGATE, R. BIXBY, V. CHVÁTAL, AND W. COOK, *On the solution of traveling salesman problems*, in Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung, International Congress of Mathematicians, 1998, pp. 645–656.
- [7] A. ATAMTÜRK AND V. NARAYANAN, *Conic mixed integer rounding cuts*, Mathematical Programming, **122** (2010), pp. 1–20.
- [8] E. BALAS, *Disjunctive programming*, in Annals of Discrete Mathematics 5: Discrete Optimization, North Holland, 1979, pp. 3–51.
- [9] E. BALAS, S. CERIA, AND G. CORNEUJOLS, *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, Mathematical Programming, **58** (1993), pp. 295–324.
- [10] E. BALAS, S. CERIA, G. CORNUÉJOLS, AND N. R. NATRAJ, *Gomory cuts revisited*, Operations Research Letters, **19** (1999), pp. 1–9.
- [11] E. BALAS AND M. PERREGAARD, *Lift-and-project for mixed 0-1 programming: recent progress*, Discrete Applied Mathematics, **123** (2002), pp. 129–154.
- [12] M.S. BAZARAA, H.D. SHERALI, AND C.M. SHETTY, *Nonlinear Programming: Theory and Algorithms*, John Wiley and Sons, New York, second ed., 1993.
- [13] E.M.L. BEALE, *Branch and bound methods for mathematical programming systems*, in Discrete Optimization II, P.L. Hammer, E.L. Johnson, and B.H. Korte, eds., North Holland Publishing Co., 1979, pp. 201–219.
- [14] E.W.L. BEALE AND J.A. TOMLIN, *Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables*, in Proceedings of the 5th International Conference on Operations Research, J. Lawrence, ed., 1969, pp. 447–454.
- [15] A. BEN-TAL AND A. NEMIROVSKI, *Lectures on Modern Convex Optimization*, SIAM, 2001. MPS/SIAM Series on Optimization.
- [16] J.F. BENDERS, *Partitioning procedures for solving mixed variable programming problems*, Numerische Mathematik, **4** (1962), pp. 238–252.
- [17] M. BÉNICHOU, J.M. GAUTHIER, P. GIRODET, G. HENTGES, G. RIBIÈRE, AND O. VINCENT, *Experiments in Mixed-Integer Linear Programming*, Mathematical Programming, **1** (1971), pp. 76–94.
- [18] L. BERTACCO, M. FISCHETTI, AND A. LODI, *A feasibility pump heuristic for general mixed-integer problems*, Discrete Optimization, **4** (2007), pp. 63–76.
- [19] T. BERTHOLD, *Primal Heuristics for Mixed Integer Programs*, Master's thesis, Technische Universität Berlin, 2006.
- [20] T. BERTHOLD AND A. GLEIXNER, *Undercover - a primal heuristic for MINLP based on sub-mips generated by set covering*, Tech. Rep. ZIB-Report 09-40, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), 2009.
- [21] D. BIENSTOCK, *Computational study of a family of mixed-integer quadratic programming problems*, Mathematical Programming, **74** (1996), pp. 121–140.

- [22] R. BIXBY AND E. ROTHBERG, *Progress in computational mixed integer programming. A look back from the other side of the tipping point*, Annals of Operations Research, **149** (2007), pp. 37–41.
- [23] P. BONAMI, *Branching strategies and heuristics in branch-and-bound for convex MINLPs*, November 2008. Presentation at IMA Hot Topics Workshop: Mixed-Integer Nonlinear Optimization: Algorithmic Advances and Applications.
- [24] P. BONAMI, L.T. BIEGLER, A.R. CONN, G. CORNUÉJOLS, I.E. GROSSMANN, C.D. LAIRD, J. LEE, A. LODI, F. MARGOT, N. SAWAYA, AND A. WÄCHTER, *An algorithmic framework for convex Mixed Integer Nonlinear Programs*, Discrete Optimization, **5** (2008), pp. 186–204.
- [25] P. BONAMI, G. CORNUÉJOLS, A. LODI, AND F. MARGOT, *A feasibility pump for Mixed Integer Nonlinear Programs*, Mathematical Programming, **119** (2009), pp. 331–352.
- [26] P. BONAMI AND J. GONÇALVES, *Heuristics for convex mixed integer nonlinear programs*, Computational Optimization and Applications. To appear DOI: 10.1007/s10589-010-9350-6.
- [27] R. BOORSTYN AND H. FRANK, *Large-scale network topological optimization*, IEEE Transactions on Communications, **25** (1977), pp. 29–47.
- [28] B. BORCHERS AND J.E. MITCHELL, *An improved branch and bound algorithm for Mixed Integer Nonlinear Programs*, Computers & Operations Research, **21** (1994), pp. 359–368.
- [29] ———, *A computational comparison of branch and bound and outer approximation algorithms for 0-1 Mixed Integer Nonlinear Programs*, Computers & Operations Research, **24** (1997), pp. 699–701.
- [30] M.R. BUSSIECK AND A. DRUD, *Sbb: A new solver for Mixed Integer Nonlinear Programming*, talk, OR 2001, Section "Continuous Optimization", 2001.
- [31] M.R. BUSSIECK, A.S. DRUD, AND A. MEERAUS, *MINLPLib – a collection of test models for Mixed-Integer Nonlinear Programming*, INFORMS Journal on Computing, **15** (2003).
- [32] R.H. BYRD, J. NOCEDAL, AND R.A. WALTZ, *KNITRO: An integrated package for nonlinear optimization*, in Large Scale Nonlinear Optimization, Springer Verlag, 2006, pp. 35–59.
- [33] I. CASTILLO, J. WESTERLUND, S. EMET, AND T. WESTERLUND, *Optimization of block layout design problems with unequal areas: A comparison of milp and minlp optimization methods*, Computers and Chemical Engineering, **30** (2005), pp. 54–69.
- [34] M.T. CEZIK AND G. IYENGAR, *Cuts for mixed 0-1 conic programming*, Mathematical Programming, **104** (2005), pp. 179–202.
- [35] V. CHVÁTAL, *Edmonds polytopes and a hierarchy of combinatorial problems*, Discrete Mathematics, **4** (1973), pp. 305–337.
- [36] M. CONFORTI, G. CORNUÉJOLS, AND G. ZAMBELLI, *Polyhedral approaches to Mixed Integer Linear Programming*, in 50 Years of Integer Programming 1958–2008, M. Jünger, T. Liebling, D. Naddef, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, eds., Springer, 2009.
- [37] G. CORNUÉJOLS, L. LIBERTI, AND G. NANNICINI, *Improved strategies for branching on general disjunctions*. To appear in Mathematical Programming A. DOI: 10.1007/s10107-009-0333-2.
- [38] R.J. DAKIN, *A tree search algorithm for mixed programming problems*, Computer Journal, **8** (1965), pp. 250–255.
- [39] E. DANNA, E. ROTHBERG, AND C. LEPAPE, *Exploring relaxation induced neighborhoods to improve MIP solutions*, Mathematical Programming, **102** (2005), pp. 71–90.
- [40] E. DOLAN AND J. MORÉ, *Benchmarking optimization software with performance profiles*, Mathematical Programming, **91** (2002), pp. 201–213.

- [41] S. DREWES, *Mixed Integer Second Order Cone Programming*, PhD thesis, Technische Universität Darmstadt, 2009.
- [42] A. S. DRUD, *CONOPT – a large-scale GRG code*, ORSA Journal on Computing, **6** (1994), pp. 207–216.
- [43] M.A. DURAN AND I. GROSSMANN, *An outer-approximation algorithm for a class of Mixed-Integer Nonlinear Programs*, Mathematical Programming, **36** (1986), pp. 307–339.
- [44] J. ECKSTEIN, *Parallel branch-and-bound algorithms for general mixed integer programming on the CM-5*, SIAM Journal on Optimization, **4** (1994), pp. 794–814.
- [45] S. ELHEDHLI, *Service System Design with Immobile Servers, Stochastic Demand, and Congestion*, Manufacturing & Service Operations Management, **8** (2006), pp. 92–97.
- [46] A. M. ELICECHE, S. M. CORVALÁN, AND P. MARTÍNEZ, *Environmental life cycle impact as a tool for process optimisation of a utility plant*, Computers and Chemical Engineering, **31** (2007), pp. 648–656.
- [47] FAIR ISAAC CORPORATION, *XPRESS-MP Reference Manual*, 2009. Release 2009.
- [48] M. FISCHETTI, F. GLOVER, AND A. LODI, *The feasibility pump*, Mathematical Programming, **104** (2005), pp. 91–104.
- [49] M. FISCHETTI AND A. LODI, *Local branching*, Mathematical Programming, **98** (2003), pp. 23–47.
- [50] M. FISCHETTI AND D. SALVAGNIN, *Feasibility pump 2.0*, Tech. Rep., University of Padova, 2008.
- [51] R. FLETCHER AND S. LEYFFER, *Solving Mixed Integer Nonlinear Programs by outer approximation*, Mathematical Programming, **66** (1994), pp. 327–349.
- [52] ———, *User manual for filterSQP*, 1998. University of Dundee Numerical Analysis Report NA-181.
- [53] A. FLORES-TLACUAHUAC AND L.T. BIEGLER, *Simultaneous mixed-integer dynamic optimization for integrated design and control*, Computers and Chemical Engineering, **31** (2007), pp. 648–656.
- [54] J.J.H. FORREST, J.P.H. HIRST, AND J.A. TOMLIN, *Practical solution of large scale mixed integer programming problems with UMPIRE*, Management Science, **20** (1974), pp. 736–773.
- [55] M.R. GAREY AND D.S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.
- [56] A. GEOFFRION, *Generalized Benders decomposition*, Journal of Optimization Theory and Applications, **10** (1972), pp. 237–260.
- [57] P.E. GILL, W. MURRAY, AND M.A. SAUNDERS, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Journal on Optimization, **12** (2002), pp. 979–1006.
- [58] R.E. GOMORY, *Outline of an algorithm for integer solutions to linear programs*, Bulletin of the American Mathematical Monthly, **64** (1958), pp. 275–278.
- [59] ———, *An algorithm for the mixed integer problem*, Tech. Rep. RM-2597, The RAND Corporation, 1960.
- [60] I. GROSSMANN, J. VISWANATHAN, A.V.R. RAMAN, AND E. KALVELAGEN, *GAMS/DICOPT: A discrete continuous optimization package*, Math. Methods Appl. Sci., **11** (2001), pp. 649–664.
- [61] O. GÜNLÜK, J. LEE, AND R. WEISMANTEL, *MINLP strengthening for separable convex quadratic transportation-cost ufl*, Tech. Rep. RC24213 (W0703-042), IBM Research Division, March 2007.
- [62] O.K. GUPTA AND A. RAVINDRAN, *Branch and bound experiments in convex nonlinear integer programming*, Management Science, **31** (1985), pp. 1533–1546.
- [63] GUROBI OPTIMIZATION, *Gurobi Optimizer Reference Manual*, 2009. Version 2.
- [64] I. HARJUNKOSKI, R. PÖRN, AND T. WESTERLUND, *MINLP: Trim-loss problem*, in Encyclopedia of Optimization, C.A. Floudas and P.M. Pardalos, eds., Springer, 2009, pp. 2190–2198.

- [65] J.-B. HIRIART-URRUTY AND C. LEMARECHAL, *Convex Analysis and Minimization Algorithms I: Fundamentals (Grundlehren Der Mathematischen Wissenschaften)*, Springer, October 1993.
- [66] IBM, *Using the CPLEX Callable Library, Version 12*, 2009.
- [67] R. JEROSLOW, *There cannot be any algorithm for integer programming with quadratic constraints*, Operations Research, **21** (1973), pp. 221–224.
- [68] N.J. JOBST, M.D. HORNIMAN, C.A. LUCAS, AND G. MITRA, *Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints*, Quantitative Finance, **1** (2001), pp. 489–501.
- [69] M. KARAMANOV AND G. CORNUÉJOLS, *Branching on general disjunctions*, tech. rep., Carnegie Mellon University, July 2005. Revised August 2009. Available at <http://integer.tepper.cmu.edu>.
- [70] J.E. KELLEY, *The cutting plane method for solving convex programs*, Journal of SIAM, **8** (1960), pp. 703–712.
- [71] M. KILINÇ, J. LINDEROTH, J. LUEDTKE, AND A. MILLER, *Disjunctive strong branching inequalities for Mixed Integer Nonlinear Programs*.
- [72] G.R. KOCIS AND I.E. GROSSMANN, *Relaxation strategy for the structural optimization of process flowcharts*, Industrial Engineering Chemical Research, **26** (1987), pp. 1869–1880.
- [73] C.D. LAIRD, L.T. BIEGLER, AND B. VAN BLOEMEN WAANDERS, *A mixed integer approach for obtaining unique solutions in source inversion of drinking water networks*, Journal of Water Resources Planning and Management, Special Issue on Drinking Water Distribution Systems Security, **132** (2006), pp. 242–251.
- [74] A.H. LAND AND A.G. DOIG, *An automatic method for solving discrete programming problems*, Econometrica, **28** (1960), pp. 497–520.
- [75] M. LEJEUNE, *A unified approach for cycle service levels*, Tech. Rep., George Washington University, 2009. Available on Optimization Online http://www.optimization-online.org/DB_HTML/2008/11/2144.html.
- [76] S. LEYFFER, *Deterministic Methods for Mixed Integer Nonlinear Programming*, PhD thesis, University of Dundee, Dundee, Scotland, UK, 1993.
- [77] ———, *User manual for MINLP-BB*, 1998. University of Dundee.
- [78] ———, *Integrating SQP and branch-and-bound for Mixed Integer Nonlinear Programming*, Computational Optimization & Applications, **18** (2001), pp. 295–309.
- [79] ———, *MacMINLP: Test problems for Mixed Integer Nonlinear Programming*, 2003. <http://www.mcs.anl.gov/~leyffer/macminlp>.
- [80] ———, *Nonlinear branch-and-bound revisited*, August 2009. Presentation at 20th International Symposium on Mathematical Programming.
- [81] L. LIBERTI, *Reformulations in mathematical programming: Symmetry*, Mathematical Programming (2010). To appear.
- [82] J. LINDEROTH AND T. RALPHS, *Noncommercial software for Mixed-Integer Linear Programming*, in Integer Programming: Theory and Practice, CRC Press Operations Research Series, 2005, pp. 253–303.
- [83] J.T. LINDEROTH AND M.W.P. SAVELSBERGH, *A computational study of search strategies in mixed integer programming*, INFORMS Journal on Computing, **11** (1999), pp. 173–187.
- [84] A. LODI, *MIP computation and beyond*, in 50 Years of Integer Programming 1958–2008, M. Jünger, T. Liebling, D. Naddef, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, eds., Springer, 2009.
- [85] H. MARCHAND AND L.A. WOLSEY, *Aggregation and mixed integer rounding to solve MIPs*, Operations Research, **49** (2001), pp. 363–371.
- [86] F. MARGOT, *Exploiting orbits in symmetric ILP*, Mathematical Programming, Series B, **98** (2003), pp. 3–21.

- [87] R.D. MCBRIDE AND J.S. YORMARK, *An implicit enumeration algorithm for quadratic integer programming*, Management Science, **26** (1980), pp. 282–296.
- [88] *Mosek ApS*, 2009. www.mosek.com.
- [89] B. MURTAGH AND M. SAUNDERS, *MINOS 5.4 user's guide*, Report SOL 83-20R, Department of Operations Research, Stanford University, 1993.
- [90] K.G. MURTY AND S.N. KABADI, *Some NP-complete problems in quadratic and nonlinear programming*, Mathematical Programming, **39** (1987), pp. 117–129.
- [91] S. NABAL AND L. SCHRAGE, *Modeling and solving nonlinear integer programming problems*. Presented at Annual AIChE Meeting, Chicago, 1990.
- [92] G. NEMHAUSER AND L.A. WOLSEY, *Integer and Combinatorial Optimization*, John Wiley and Sons, New York, 1988.
- [93] G.L. NEMHAUSER, M.W.P. SAVELSBERGH, AND G.C. SIGISMONDI, *MINTO, a Mixed INTeger Optimizer*, Operations Research Letters, **15** (1994), pp. 47–58.
- [94] J. NOCEDAL AND S.J. WRIGHT, *Numerical Optimization*, Springer-Verlag, New York, second ed., 2006.
- [95] J. OSTROWSKI, J. LINDEROTH, F. ROSSI, AND S. SMRIGLIO, *Orbital branching*, Mathematical Programming, **126** (2011), pp. 147–178.
- [96] I. QUESADA AND I.E. GROSSMANN, *An LP/NLP based branch-and-bound algorithm for convex MINLP optimization problems*, Computers and Chemical Engineering, **16** (1992), pp. 937–947.
- [97] D.E. RAVEMARK AND D.W.T. RIPPIN, *Optimal design of a multi-product batch plant*, Computers & Chemical Engineering, **22** (1998), pp. 177 – 183.
- [98] N. SAWAYA, *Reformulations, relaxations and cutting planes for generalized disjunctive programming*, PhD thesis, Chemical Engineering Department, Carnegie Mellon University, 2006.
- [99] N. SAWAYA, C.D. LAIRD, L.T. BIEGLER, P. BONAMI, A.R. CONN, G. CORNUÉJOLS, I. E. GROSSMANN, J. LEE, A. LODI, F. MARGOT, AND A. WÄCHTER, *CMU-IBM open source MINLP project test set*, 2006. <http://egon.cheme.cmu.edu/ibm/page.htm>.
- [100] A. SCHRIJVER, *Theory of Linear and Integer Programming*, Wiley, Chichester, 1986.
- [101] R. STUBBS AND S. MEHROTRA, *A branch-and-cut method for 0-1 mixed convex programming*, Mathematical Programming, **86** (1999), pp. 515–532.
- [102] M. TÜRKAY AND I.E. GROSSMANN, *Logic-based minlp algorithms for the optimal synthesis of process networks*, Computers & Chemical Engineering, **20** (1996), pp. 959 – 978.
- [103] R.J. VANDERBEI, *LOQO: An interior point code for quadratic programming*, Optimization Methods and Software (1998).
- [104] A. VECCHIETTI AND I.E. GROSSMANN, *LOGMIP: a disjunctive 0-1 non-linear optimizer for process system models*, Computers and Chemical Engineering, **23** (1999), pp. 555 – 565.
- [105] J. VISWANATHAN AND I.E. GROSSMANN, *A combined penalty function and outer-approximation method for MINLP optimization*, Computers and Chemical Engineering, **14** (1990), pp. 769–782.
- [106] A. WÄCHTER, *Some recent advances in Mixed-Integer Nonlinear Programming*, May 2008. Presentation at the SIAM Conference on Optimization.
- [107] A. WÄCHTER AND L.T. BIEGLER, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Mathematical Programming, **106** (2006), pp. 25–57.
- [108] R. WALTZ, *Current challenges in nonlinear optimization*, 2007. Presentation at San Diego Supercomputer Center: CIEG Spring Orientation Workshop, available at www.sdsc.edu/us/training/workshops/2007sac_studentworkshop/docs/SDSC07.ppt.

- [109] T. WESTERLUND, H.I. HARJUNKOSKI, AND R. PÖRN, *An extended cutting plane method for solving a class of non-convex minlp problems*, Computers and Chemical Engineering, **22** (1998), pp. 357–365.
- [110] T. WESTERLUND AND K. LUNDQVIST, *Alpha-ECP, version 5.101. an interactive minlp-solver based on the extended cutting plane method*, in Updated version of Report 01-178-A, Process Design Laboratory, Abo Akademi University, 2005.
- [111] T. WESTERLUND AND F. PETTERSSON, *A cutting plane method for solving convex MINLP problems*, Computers and Chemical Engineering, **19** (1995), pp. s131–s136.
- [112] T. WESTERLUND AND R. PÖRN, . *a cutting plane method for minimizing pseudo-convex functions in the mixed integer case*, Computers and Chemical Engineering, **24** (2000), pp. 2655–2665.
- [113] L.A. WOLSEY, *Integer Programming*, John Wiley and Sons, New York, 1998.
- [114] Y. ZHU AND T. KUNO, *A disjunctive cutting-plane-based branch-and-cut algorithm for 0-1 mixed-integer convex nonlinear programs*, Industrial and Engineering Chemistry Research, **45** (2006), pp. 187–196.

SUBGRADIENT BASED OUTER APPROXIMATION FOR MIXED INTEGER SECOND ORDER CONE PROGRAMMING*

SARAH DREWES[†] AND STEFAN ULRICH[‡]

Abstract. This paper deals with outer approximation based approaches to solve mixed integer second order cone programs. Thereby the outer approximation is based on subgradients of the second order cone constraints. Using strong duality of the subproblems that are solved during the algorithm, we are able to determine subgradients satisfying the KKT optimality conditions. This enables us to extend convergence results valid for continuously differentiable mixed integer nonlinear problems to subdifferentiable constraint functions. Furthermore, we present a version of the branch-and-bound based outer approximation that converges when relaxing the convergence assumption that every SOCP satisfies the Slater constraint qualification. We give numerical results for some application problems showing the performance of our approach.

Key words. Mixed Integer Nonlinear Programming, Second Order Cone Programming, Outer Approximation.

AMS(MOS) subject classifications. 90C11.

1. Introduction. Mixed Integer Second Order Cone Programs (MIS-OCP) can be formulated as

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \succeq 0 \\ & (x)_j \in [l_j, u_j] \quad (j \in J), \\ & (x)_j \in \mathbb{Z} \quad (j \in J), \end{aligned} \tag{1.1}$$

where $c = (c_1^T, \dots, c_{noc}^T)^T \in \mathbb{R}^n$, $A = (A_1, \dots, A_{noc}) \in \mathbb{R}^{m,n}$, with $c_i \in \mathbb{R}^{k_i}$ and $A_i \in \mathbb{R}^{m,k_i}$ for $i \in \{1, \dots, noc\}$ and $\sum_{i=1}^{noc} k_i = n$. Furthermore, $b \in \mathbb{R}^m$, $(x)_j$ denotes the j -th component of x , $l_j, u_j \in \mathbb{R}$ for $j \in J$ and $J \subset \{1, \dots, n\}$ denotes the integer index set. Here, $x \succeq 0$ for $x = (x_1^T, \dots, x_{noc}^T)^T$ with $x_i \in \mathbb{R}^{k_i}$ for $i \in \{1, \dots, noc\}$ denotes that

$$x \in \mathcal{K}, \quad \mathcal{K} := \mathcal{K}_1 \times \dots \times \mathcal{K}_{noc},$$

where

$$\mathcal{K}_i := \{x_i = (x_{i0}, x_{i1}^T)^T \in \mathbb{R} \times \mathbb{R}^{k_i-1} : \|x_{i1}\|_2 \leq x_{i0}\}$$

*Research partially supported by the German Research Foundation (DFG) within the SFB 805 and by the state of Hesse within the LOEWE-Center AdRIA.

[†]Research Group Nonlinear Optimization, Department of Mathematics, Technische Universität Darmstadt, Germany.

[‡]Research Group Nonlinear Optimization, Department of Mathematics, Technische Universität Darmstadt, Germany.

is the second order cone of dimension k_i . Mixed integer second order cone problems have various applications in finance or engineering, for example turbine balancing problems, cardinality-constrained portfolio optimization (cf. Bertsimas and Shioda in [17] or Vielma et al. in [10]) or the problem of finding a minimum length connection network also known as the Euclidean Steiner Tree Problem (ESTP) (cf. Fampa, Maculan in [15]).

Available convex MINLP solvers like **BONMIN** [22] by Bonami et al. or **Filmint** [25] by Abhishek et al. are in general not applicable for (1.1), since the occurring second order cone constraints are not continuously differentiable.

Branch-and-cut methods for convex mixed 0-1 problems have been discussed by Stubbs and Mehrotra in [2] and [9] which can be applied to solve (1.1), if the integer variables are binary. In [5] Çezik and Iyengar discuss cuts for general self-dual conic programming problems and investigate their applications on the maxcut and the traveling salesman problem. Atamtürk and Narayanan present in [12] integer rounding cuts for conic mixed-integer programming by investigating polyhedral decompositions of the second order cone conditions and in [11] the authors discuss lifting for mixed integer conic programming, where valid inequalities for mixed-integer feasible sets are derived from suitable subsets.

One article dealing with non-differentiable functions in the context of outer approximation approaches for MINLP is [1] by Fletcher and Leyffer, where the authors prove convergence of outer approximation algorithms for non-smooth penalty functions. The only article dealing with outer approximation techniques for MISOCPs is [10] by Vielma et al., which is based on Ben-Tal and Nemirovskii's polyhedral approximation of the second order cone constraints [13]. Thereby, the size of the outer approximation grows when strengthening the precision of the approximation. This precision and thus the entire outer approximation is chosen in advance, whereas the approximation presented here is strengthened iteratively in order to guarantee convergence of the algorithm.

In this paper we present a hybrid branch-and-bound based outer approximation approach for MISOCPs. The approach is based on the branch-and-bound based outer approximation approach for continuously differentiable constraints – as proposed by Bonami et al. in [8] on the basis of Fletcher and Leyffer [1] and Quesada and Grossmann [3]. The idea is to iteratively compute integer feasible solutions of a (sub)gradient based linear outer approximation of (1.1) and to tighten this outer approximation by solving nonlinear continuous problems.

Thereby linear outer approximations based on subgradients satisfying the Karush Kuhn Tucker (KKT) optimality conditions of the occurring SOCP problems enable us to extend the convergence result for continuously differentiable constraints to subdifferentiable second order cone constraints.

Thus, in contrast to [10], the subgradient based approximation induces convergence of any classical outer approximation based approach under the

specified assumptions. We also present an adaption of the algorithm that converges even if one of these convergence assumptions is violated.

In numerical experiments we show the applicability of the algorithm and compare it to a nonlinear branch-and-bound approach.

2. Preliminaries. In the following $\text{int}(\mathcal{K}_i)$ denotes the interior of the cone \mathcal{K}_i , i.e. those vectors x_i satisfying $x_{i0} > \|x_{i1}\|$, $\text{bd}(\mathcal{K}_i)$ denotes the boundary of \mathcal{K}_i , i.e. those vectors x_i satisfying $x_{i0} = \|x_{i1}\|$. By $\|\cdot\|$ we denote the Euclidean norm. Assume $g : \mathbb{R}^n \mapsto \mathbb{R}$ is a convex and subdifferentiable function on \mathbb{R}^n . Then due to the convexity of g , the inequality $g(x) \geq g(\bar{x}) + \xi^T(x - \bar{x})$ holds for all $\bar{x}, x \in \mathbb{R}^n$ and every subgradient $\xi \in \partial g(\bar{x})$ – see for example [19]. Thus, we obtain a linear outer approximation of the region $\{x : g(x) \leq 0\}$ applying constraints of the form

$$g(\bar{x}) + \xi^T(x - \bar{x}) \leq 0. \quad (2.1)$$

In the case of (1.1), the feasible region is described by constraints

$$g_i(x) := -x_{i0} + \|x_{i1}\| \leq 0, \quad i = 1, \dots, noc, \quad (2.2)$$

where $g_i(x)$ is differentiable on $\mathbb{R}^n \setminus \{x : \|x_{i1}\| = 0\}$ with $\nabla g_i(x_i) = (-1, \frac{x_{i1}^T}{\|x_{i1}\|})$ and subdifferentiable if $\|x_{i1}\| = 0$. The following lemma gives a detailed description of the subgradients of (2.2).

LEMMA 2.1. *The convex function $g_i(x_i) := -x_{i0} + \|x_{i1}\|$ is subdifferentiable in $x_i = (x_{i0}, x_{i1}^T)^T = (a, 0^T)^T$, $a \in \mathbb{R}$, with $\partial g_i((a, 0^T)^T) = \{\xi = (\xi_0, \xi_1^T)^T, \xi_0 \in \mathbb{R}, \xi_1 \in \mathbb{R}^{k_i-1} : \xi_0 = -1, \|\xi_1\| \leq 1\}$.*

Proof. Follows from the subgradient inequality in $(a, 0^T)^T$. \square

The following lemma investigates a complementarity constraint on two elements of the second order cone that is used in the subsequent sections.

LEMMA 2.2. *Assume \mathcal{K} is the second order cone of dimension k and $x = (x_0, x_1^T)^T \in \mathcal{K}$, $s = (s_0, s_1^T)^T \in \mathcal{K}$ satisfy the condition $x^T s = 0$, then*

1. $x \in \text{int}(\mathcal{K}) \Rightarrow s = (0, \dots, 0)^T$,
2. $x \in \text{bd}(\mathcal{K}) \setminus \{0\} \Rightarrow s \in \text{bd}(\mathcal{K})$ and $\exists \gamma \geq 0 : s = \gamma(x_0, -x_1^T)$.

Proof. 1.: Assume $\|x_1\| > 0$ and $s_0 > 0$. Due to $x_0 > \|\bar{x}_1\|$ it holds that $s^T x = s_0 x_0 + s_1^T x_1 > s_0 \|x_1\| + s_1^T x_1 \geq s_0 \|x_1\| - \|s_1\| \|x_1\|$. Then $x^T s = 0$ can only be true, if $s_0 \|x_1\| - \|s_1\| \|x_1\| < 0 \Leftrightarrow s_0 < \|s_1\|$ which contradicts $s \in \mathcal{K}$. Thus, $s_0 = 0 \Rightarrow s = (0, \dots, 0)^T$. If $\|x_1\| = 0$, then $s_0 = 0$ follows directly from $x_0 > 0$.

2.: Due to $x_0 = \|x_1\|$, we have $s^T x = 0 \Leftrightarrow -s_1^T x_1 = s_0 \|x_1\|$. Since $s_0 \geq \|s_1\| \geq 0$ we have $-s_1^T x_1 = s_0 \|x_1\| \geq \|x_1\| \|s_1\|$. Cauchy -Schwarz's inequality yields $-s_1^T x_1 = \|x_1\| \|s_1\|$ which implies both $s_1 = -\gamma x_1$, $\gamma \in \mathbb{R}$ and $s_0 = \|s_1\|$. It follows that $-x_1^T s_1 = \gamma x_1^T x_1 \geq 0$. Together with $s_0 = \|s_1\|$ and $\|x_1\| = x_0$ we get that there exists $\gamma \geq 0$, such that $s_1 = (\|-\gamma x_1\|, -\gamma x_1^T)^T = \gamma(x_0, -x_1^T)^T$. \square

We make the following assumptions:

- A1.** The set $\{x : Ax = b, x_J \in [l, u]\}$ is bounded.
- A2.** Every nonlinear subproblem ($NLP(x_J^k)$) that is obtained from (1.1) by fixing the integer variables to the value x_J^k has nonempty interior (Slater constraint qualification).

These assumptions comply with the assumptions made by Fletcher and Leyffer in [1] as follows. We drop the assumption of continuous differentiability of the constraint functions, but we assume a constraint qualification inducing strong duality instead of an arbitrary constraint qualification which suffices in the differentiable case.

REMARK. A2 might be expected as a strong assumption, since it is violated as soon as a leading cone variable x_{i0} is fixed to zero. In that case, all variables belonging to that cone can be eliminated and the Slater condition may hold now for the reduced problem. Moreover, at the end of Section 5, we present an enhancement of the algorithm that converges even if assumption A2 is violated.

3. Feasible nonlinear subproblems. For a given integer configuration x_J^k , we define the SOCP subproblem

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b, \\ & x \succeq 0, \\ & x_J = x_J^k. \end{aligned} \tag{NLP(x_J^k)}$$

The dual of ($NLP(x_J^k)$), in the sense of Nesterov and Nemirovskii [18] or Alizadeh and Goldfarb [7], is given by

$$\begin{aligned} \max \quad & (b^T, x_J^{k,T})y \\ \text{s.t.} \quad & (A^T, I_J^T)y + s = c, \\ & s \succeq 0, \end{aligned} \tag{NLP(x_J^k)-D}$$

where $I_J = ((I_J)_1, \dots, (I_J)_{noc})$ denotes the matrix mapping x to the integer variables x_J where $(I_J)_i \in \mathbb{R}^{|J|, k_i}$ is the block of columns of I_J associated with the i -th cone of dimension k_i . We define

$$\begin{aligned} I_0(\bar{x}) &:= \{i : \bar{x}_i = (0, \dots, 0)^T\}, \\ I_a(\bar{x}) &:= \{i : g_i(\bar{x}) = 0, \bar{x}_i \neq (0, \dots, 0)^T\}, \end{aligned} \tag{3.1}$$

where $I_a(\bar{x})$ is the index set of active conic constraints that are differentiable in \bar{x} and $I_0(\bar{x})$ is the index set of active constraints that are subdifferentiable in \bar{x} . The crucial point in an outer approximation approach is to tighten the outer approximation problem such that the integer assignment of the last solution is cut off. Assume x_J^k is this last solution. Then we will show later that those subgradients in $\partial g_i(\bar{x})$ that satisfy the KKT conditions in the solution \bar{x} of ($NLP(x_J^k)$) give rise to linearizations with this

tightening property. Hence, we show now, how to choose elements $\bar{\xi}_i$ in the subdifferentials $\partial g_i(\bar{x})$ for $i \in \{1, \dots, noc\}$ that satisfy the KKT conditions

$$\begin{aligned} c_i + (A_i^T, (I_J)_i^T)\bar{\mu} + \bar{\lambda}_i \bar{\xi}_i &= 0, \quad i \in I_0(\bar{x}), \\ c_i + (A_i^T, (I_J)_i^T)\bar{\mu} + \bar{\lambda}_i \nabla g_i(\bar{x}_i) &= 0, \quad i \in I_a(\bar{x}), \\ c_i + (A_i^T, (I_J)_i^T)\bar{\mu} &= 0, \quad i \notin I_0(\bar{x}) \cup I_a(\bar{x}) \end{aligned} \quad (3.2)$$

in the solution \bar{x} of $(NLP(x_J^k))$ with appropriate Lagrange multipliers $\bar{\mu}$ and $\bar{\lambda} \geq 0$. This step is not necessary if the constraint functions are continuously differentiable, since $\partial g_i(\bar{x})$ then contains only one element: the gradient $\nabla g_i(\bar{x})$.

LEMMA 3.1. *Assume A1 and A2. Let \bar{x} solve $(NLP(x_J^k))$ and let (\bar{s}, \bar{y}) be the corresponding dual solution of $(NLP(x_J^k)-D)$. Then there exist Lagrange multipliers $\bar{\mu} = -\bar{y}$ and $\bar{\lambda}_i \geq 0$ ($i \in I_0 \cup I_a$) that solve the KKT conditions (3.2) in \bar{x} with subgradients*

$$\bar{\xi}_i = \begin{pmatrix} -1 \\ -\frac{\bar{s}_{i1}}{\bar{s}_{i0}} \end{pmatrix}, \text{ if } \bar{s}_{i0} > 0, \quad \bar{\xi}_i = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \text{ if } \bar{s}_{i0} = 0 \quad (i \in I_0(\bar{x})).$$

Proof. A1 and A2 guarantee the existence of a primal-dual solution $(\bar{x}, \bar{s}, \bar{y})$ satisfying the primal dual optimality system (cf. Alizadeh and Goldfarb [7])

$$c_i - (A_i^T, (I_J)_i^T)\bar{y} = \bar{s}_i, \quad i = 1, \dots, noc, \quad (3.3)$$

$$A\bar{x} = b, \quad I_J\bar{x} = x_J^k, \quad (3.4)$$

$$\bar{x}_{i0} \geq \|\bar{x}_{i1}\|, \quad \bar{s}_{i0} \geq \|\bar{s}_{i1}\|, \quad i = 1, \dots, noc, \quad (3.5)$$

$$\bar{s}_i^T \bar{x}_i = 0, \quad i = 1, \dots, noc. \quad (3.6)$$

Since $(NLP(x_J^k))$ is convex and due to A2, there also exist Lagrange multipliers $\bar{\mu} \in \mathbb{R}^m$, $\bar{\lambda} \in \mathbb{R}^{noc}$, such that \bar{x} satisfies the KKT-conditions (3.2) with elements $\bar{\xi}_i \in \partial g_i(\bar{x})$. We now compare both optimality systems to each other.

First, we consider $i \notin I_0 \cup I_a$ and thus $\bar{x}_i \in \text{int}(\mathcal{K}_i)$. Lemma 2.2, part 1 induces $\bar{s}_i = (0, \dots, 0)^T$. Conditions (3.3) for $i \notin I_0 \cup I_a$ are thus equal to $c_i - (A_i^T, (I_J)_i^T)\bar{y} = 0$ and thus $\bar{\mu} = -\bar{y}$ satisfies the KKT-condition (3.2) for $i \notin I_0 \cup I_a$.

Next we consider $i \in I_a(\bar{x})$, where $x_i \in \text{bd}(\mathcal{K}) \setminus \{0\}$. Lemma 2.2, part 2 yields

$$\bar{s}_i = \begin{pmatrix} \|-\gamma \bar{x}_{i1}\| \\ -\gamma \bar{x}_{i1} \end{pmatrix} = \gamma \begin{pmatrix} \bar{x}_{i0} \\ -\bar{x}_{i1} \end{pmatrix} \quad (3.7)$$

for $i \in I_a(\bar{x})$. Inserting $\nabla g_i(\bar{x}) = (-1, \frac{\bar{x}_{i1}^T}{\|\bar{x}_{i1}\|})^T$ for $i \in I_a$ into (3.2) yields the existence of $\lambda_i \geq 0$ such that

$$c_i + (A_i^T, (I_J)_i^T)\mu = \lambda_i \begin{pmatrix} 1 \\ -\frac{\bar{x}_{i1}}{\|\bar{x}_{i1}\|} \end{pmatrix}, \quad i \in I_a(\bar{x}). \quad (3.8)$$

Insertion of (3.7) into (3.3) and comparison with (3.8) yields the existence of $\gamma \geq 0$ such that $\bar{\mu} = -\bar{y}$ and $\bar{\lambda}_i = \gamma \bar{x}_{i0} = \gamma \|\bar{x}_{i1}\| \geq 0$ satisfy the KKT-conditions (3.2) for $i \in I_a(\bar{x})$.

For $i \in I_0(\bar{x})$, condition (3.2) is satisfied by $\mu \in \mathbb{R}^m$, $\bar{\lambda}_i \geq 0$ and subgradients ξ_i of the form $\bar{\xi}_i = (-1, v^T)^T$, $\|v\| \leq 1$. Since $\bar{\mu} = -\bar{y}$ satisfies (3.2) for $i \notin I_0$, we look for a suitable v and $\bar{\lambda}_i \geq 0$ satisfying $c_i - (A_i^T, (I_J)^T)\bar{y} = \bar{\lambda}_i(1, -v^T)^T$ for $i \in I_0(\bar{x})$. Comparing the last condition with (3.3) yields that if $\|\bar{s}_{i1}\| > 0$, then $\bar{\lambda}_i = \bar{s}_{i0}, -v = \frac{\bar{s}_{i1}}{\bar{s}_{i0}}$ satisfy condition (3.2) for $i \in I_0(\bar{x})$. Since $\bar{s}_{i0} \geq \|\bar{s}_{i1}\|$ we obviously have $\bar{\lambda}_i \geq 0$ and $\|v\| = \|\frac{\bar{s}_{i1}}{\bar{s}_{i0}}\| = \frac{1}{\bar{s}_{i0}}\|\bar{s}_{i1}\| \leq 1$. If $\|\bar{s}_{i1}\| = 0$, the required condition (3.2) is satisfied by $\bar{\lambda}_i = \bar{s}_{i0}, -v = (0, \dots, 0)^T$. \square

4. Infeasible nonlinear subproblems. If the nonlinear program $(NLP(x_J^k))$ is infeasible for x_J^k , the algorithm solves a feasibility problem of the form

$$\begin{aligned} & \min u \\ \text{s.t. } & Ax = b, \\ & -x_{i0} + \|x_{i1}\| \leq u, \quad i = 1, \dots, noc, \\ & u \geq 0, \\ & x_J = x_J^k. \end{aligned} \tag{F(x_J^k)}$$

It has the property that the optimal solution (\bar{x}, \bar{u}) minimizes the maximal violation of the conic constraints. The dual program of $(F(x_J^k))$ is

$$\begin{aligned} \max & \quad (b^T, x_J^{k,T})y \\ \text{s.t. } & -(A^T, I_J^T)y + s = 0, \\ & s_u + \sum_{i=1}^{noc} s_{i0} = 1, \\ & \|s_{i1}\| \leq s_{i0}, \quad i = 1, \dots, noc, \\ & s_u \geq 0. \end{aligned} \tag{(F(x_J^k)-D)}$$

We define the index sets of active constraints in a solution (\bar{x}, \bar{u}) of $(F(x_J^k))$,

$$\begin{aligned} I_F := I_F(\bar{x}) &:= \{i \in \{1, \dots, noc\} : -\bar{x}_{i0} + \|\bar{x}_{i1}\| = \bar{u}\}, \\ I_{F0} := I_{F0}(\bar{x}) &:= \{i \in I_F : \|\bar{x}_{i1}\| = 0\}, \\ I_{F1} := I_{F1}(\bar{x}) &:= \{i \in I_F : \|\bar{x}_{i1}\| \neq 0\}. \end{aligned} \tag{4.1}$$

One necessity for convergence of the outer approximation approach is the following. Analogously to the feasible case, the solution of the feasibility problem $(F(x_J^k))$ must tighten the outer approximation such that the current integer assignment x_J^k is no longer feasible for the linear outer approximation. For this purpose, we identify subgradients $\xi_i \in \partial g_i(\bar{x})$ at the solution (\bar{u}, \bar{x}) of $(F(x_J^k))$ that satisfy the KKT conditions of $(F(x_J^k))$

$$A_i^T \mu_A + (I_J)_i^T \mu_J = 0, i \notin I_F, \quad (4.2)$$

$$\nabla g_i(\bar{x}_i) \lambda_{g_i} + A_i^T \mu_A + (I_J)_i^T \mu_J = 0, i \in I_{F1}, \quad (4.3)$$

$$\xi_i \lambda_{g_i} + A_i^T \mu_A + (I_J)_i^T \mu_J = 0, i \in I_{F0}, \quad (4.4)$$

$$\sum_{i \in I_F} (\lambda_g)_i = 1. \quad (4.5)$$

LEMMA 4.1. Assume A1 and A2 hold. Let (\bar{x}, \bar{u}) solve $(F(x_J^k))$ with $\bar{u} > 0$ and let (\bar{s}, \bar{y}) be the solution of its dual program $(F(x_J^k)\text{-}D)$. Then there exist Lagrange multipliers $\bar{\mu} = -\bar{y}$ and $\bar{\lambda}_i \geq 0$ ($i \in I_F$) that solve the KKT conditions in (\bar{x}, \bar{u}) with subgradients

$$\xi_i = \begin{pmatrix} -1 \\ -\frac{\bar{s}_{i1}}{\bar{s}_{i0}} \end{pmatrix}, \text{ if } \bar{s}_{i0} > 0, \quad \xi_i = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \text{ if } \bar{s}_{i0} = 0 \quad (4.6)$$

for $i \in I_{F0}(\bar{x})$.

Proof. Since $(F(x_J^k))$ has interior points, there exist Lagrange multipliers $\mu \in \mathbb{R}^m$, $\lambda \geq 0$, such that optimal solution (\bar{x}, \bar{u}) of $(F(x_J^k))$ satisfies the KKT-conditions (4.2) - (4.5) with $\xi_i \in \partial g_i(\bar{x}_i)$ plus the feasibility conditions. We already used the complementary conditions for $\bar{u} > 0$ and the inactive constraints. Due to the nonempty interior of $(F(x_J^k))$, (\bar{x}, \bar{u}) satisfies also the primal-dual optimality system

$$Ax = b,$$

$$u \geq 0,$$

$$-A_i^T y_A - (I_J)_i^T y_J = s_i, i = 1, \dots, noc, \quad (4.7)$$

$$x_{i0} + u \geq \|\bar{x}_{i1}\|, \sum_{i=1}^{noc} s_{i0} = 1, \quad (4.8)$$

$$s_{i0} \geq \|\bar{s}_{i1}\|, i = 1, \dots, noc, \quad (4.9)$$

$$s_{i0}(x_{i0} + u) + s_{i1}^T x_{i1} = 0, i = 1, \dots, noc, \quad (4.10)$$

where we again used complementarity for $\bar{u} > 0$.

First we investigate $i \notin I_F$. In this case $\bar{x}_{i0} + \bar{u} > \|\bar{x}_{i1}\|$ induces $s_i = (0, \dots, 0)^T$ (cf. Lemma 2.2, part 1). Thus, the KKT conditions (4.2) are satisfied by $\mu_A = -y_A$ and $\mu_J = -y_J$.

Next, we consider $i \in I_{F1}$ for which by definition $\bar{x}_{i0} + \bar{u} = \|\bar{x}_{i1}\| > 0$ holds. Lemma 2.2, part 2 states that there exists $\gamma \geq 0$ with $s_{i0} = \gamma(\bar{x}_{i0} + \bar{u}) = \gamma\|\bar{x}_{i1}\|$ and $s_{i1} = -\gamma\bar{x}_{i1}$. Insertion into (4.7) yields

$$-A_i^T y_A - (I_J)_i^T y_J + \gamma\|\bar{x}_{i1}\| \begin{pmatrix} -1 \\ \frac{\bar{x}_{i1}}{\|\bar{x}_{i1}\|} \end{pmatrix} = 0, i \in I_{F1}.$$

Since $\nabla g_i(\bar{x}_i) = (-1, \frac{\bar{x}_{i1}^T}{\|\bar{x}_{i1}\|})^T$, we obtain that the KKT-condition (4.3) is satisfied by $\mu_A = -y_A$, $\mu_J = -y_J$ and $\lambda_{g_i} = s_{i0} = \gamma\|\bar{x}_{i1}\| \geq 0$.

Finally, we investigate $i \in I_{F0}$, where $\bar{x}_{i0} + \bar{u} = \|\bar{x}_{i1}\| = 0$. Since $\mu_A = -y_A$, $\mu_J = -y_J$ satisfy the KKT-conditions for $i \notin I_{F0}$, we derive a subgradient ξ_i that satisfies (4.4) with that choice. In analogy to Lemma 3.1 from Section 3 we derive that $\xi_i = (-1, \xi_{i1}^T)^T$ with $\xi_{i1} = \frac{-s_{i1}}{s_{i0}}$, if $s_{i0} > 0$ and $\xi_{i1} = 0$ otherwise, are suitable together with $\lambda_i = s_{i0} \geq 0$. \square

5. The algorithm. Let $T \subset \mathbb{R}^n$ contain solutions of nonlinear subproblems ($NLP(x_J^k)$) and let $S \subset \mathbb{R}^n$ contain solutions of feasibility problems ($F(x_J^k)$). We build a linear outer approximation of (1.1) based on subgradient based linearizations of the form (2.1). Thereby we use the subgradients specified in Lemma 3.1 and 4.1. This gives rise to the mixed integer linear outer approximation problem

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & c^T x < c^T \bar{x}, \bar{x} \in T, \bar{x}_J \in \mathbb{Z}^{|J|}, \\ & -\|\bar{x}_{i1}\| x_{i0} + \bar{x}_{i1}^T x_{i1} \leq 0, \quad i \in I_a(\bar{x}), \bar{x} \in T, \\ & -\|\bar{x}_{i1}\| x_{i0} + \bar{x}_{i1}^T x_{i1} \leq 0, \quad i \in I_{F1}(\bar{x}) \bar{x} \in S, \\ & -x_{i0} \leq 0, \quad i \in I_0(\bar{x}), \bar{s}_{i0} = 0, \bar{x} \in T, \quad (\text{MIP}(T,S)) \\ & -x_{i0} - \frac{1}{\bar{s}_{i0}} \bar{s}_{i1}^T x_{i1} \leq 0, \quad i \in I_0(\bar{x}), \bar{s}_{i0} > 0, \bar{x} \in T, \\ & -x_{i0} - \frac{1}{\bar{s}_{i0}} \bar{s}_{i1}^T x_{i1} \leq 0, \quad i \in I_{F0}(\bar{x}), \bar{s}_{i0} > 0, \bar{x} \in S, \\ & -x_{i0} \leq 0, \quad i \in I_{F0}(\bar{x}), \bar{s}_{i0} = 0, \bar{x} \in S, \\ & x_j \in [l_j, u_j], \quad (j \in J) \\ & x_j \in \mathbb{Z}, \quad (j \in J). \end{aligned}$$

The idea of outer approximation based algorithms is to use such a linear outer approximation ($\text{MIP}(T,S)$) of the original problem (1.1) to produce integer assignments. For each integer assignment the nonlinear subproblem ($NLP(x_J^k)$) is solved generating feasible solutions for (1.1) as long as ($NLP(x_J^k)$) is feasible. We define nodes N^k consisting of lower and upper bounds on the integer variables that can be interpreted as branch-and-bound nodes for (1.1) as well as ($\text{MIP}(T,S)$). We define the following problems associated with N^k :

$(MISOC^k)$	mixed integer SOCP with bounds of N^k
(SOC^k)	continuous relaxation of $(MISOC^k)$
$(MIP^k(T, S))$	MIP outer approximation of $(MISOC^k)$
$(LP^k(T, S))$	continuous relaxation of $(MIP^k(T, S))$

Thus, if $(LP^k(T, S))$ is infeasible, (SOC^k) , $(MISOC^k)$ and $(MIP^k(T, S))$ are also infeasible. The optimal objective function value of $(LP^k(T, S))$ is less or equal than the optimal objective function values of $(MIP^k(T, S))$ and (SOC^k) respectively, and these are less or equal than the optimal objective function value of $(MISOC^k)$. Thus, the algorithm stops searching the subtree of N^k either if the problem itself or its outer approximation becomes infeasible or if the objective function

value of $(MIP^k(T, S))$ exceeds the optimal function value of a known feasible solution of (1.1). The latter case is expressed in the condition $c^T x < c^T \bar{x}$, $\forall \bar{x} \in T$ in $(MIP^k(T, S))$. The following hybrid algorithm integrates branch-and-bound and the outer approximation approach as proposed by Bonami et al. in [8] for convex differentiable MINLPs.

Algorithm 1 HYBRID OA/B-A-B FOR (1.1)

Input: Problem (1.1)

Output: Optimal solution x^* or indication of infeasibility.

Initialization: $CUB := \infty$, solve (SOC^0) with solution x^0 ,

if $((SOC^0))$ infeasible **STOP**, problem infeasible
else set $S = \emptyset$, $T = \{x^0\}$ and solve $(MIP(T, S))$
endif

1. **if** $((MIP(T, S))$ infeasible **STOP**, problem infeasible

else solution $x^{(1)}$ found:

if $(NLP(x_J^{(1)}))$ feasible
compute solution \bar{x} of $(NLP(x_J^{(1)}))$, $T := T \cup \{\bar{x}\}$,
if $(c^T \bar{x} < CUB)$ $CUB = c^T \bar{x}$, $x^* = \bar{x}$ **endif**.
else compute solution \bar{x} of $F(x_J^{(1)})$, $S := S \cup \{\bar{x}\}$.
endif

endif

$Nodes := \{N^0 = (lb^0 = l, ub^0 = u)\}$, $ll := 0$, $L := 10$, $i := 0$

2. **while** $Nodes \neq \emptyset$ **do** select N^k from $Nodes$, $Nodes := Nodes \setminus N^k$

2a. **if** $(ll = 0 \bmod L)$ solve (SOC^k)

if $((SOC^k))$ feasible: solution \bar{x} , $T := T \cup \{\bar{x}\}$
if $(x_J$ integer):
if $(c^T \bar{x} < CUB)$ $CUB = c^T \bar{x}$, $x^* = \bar{x}$ **endif**
go to 2.

endif

else go to 2.

endif

endif

2b. solve $(LP^k(T, S))$ with solution x^k

while $((LP^k(T, S))$ feasible) & $(x_J^k$ integer) & $(c^T x^k < CUB)$

if $(NLP(x_J^k))$ feasible with solution \bar{x} $T := T \cup \{\bar{x}\}$

if $(c^T \bar{x} < CUB)$ $CUB = c^T \bar{x}$, $x^* = \bar{x}$ **endif**

else solve $F(x_J^k)$ with solution \bar{x} , $S := S \cup \{\bar{x}\}$

endif

compute solution x^k of updated $(LP^k(T, S))$

endwhile

2c. **if** $(c^T x^k < CUB)$ branch on variable $x_J^k \notin \mathbb{Z}$,

create $N^{i+1} = N^k$, with $ub_j^{i+1} = \lfloor x_j^k \rfloor$,

create $N^{i+2} = N^k$, with $lb_j^{i+2} = \lceil x_j^k \rceil$,

```

    set  $i = i + 2$ ,  $ll = ll + 1$ .
endif
endwhile

```

Note that if $L = 1$, x^k is set to \bar{x} and Step 2b is omitted, Step 2 performs a nonlinear branch-and-bound search. If $L = \infty$ Algorithm 1 resembles an LP/NLP-based branch-and-bound algorithm. Convergence of the outer approximation approach in case of continuously differentiable constraint functions was shown in [1], Theorem 2. We now state convergence of Algorithm 1 for subdifferentiable SOCP constraints.

For this purpose, we first prove that the last integer assignment x_J^k is infeasible in the outer approximation conditions induced by the solution of a feasible subproblem ($NLP(x_J^k)$).

LEMMA 5.1. *Assume A1 and A2 hold. If ($NLP(x_J^k)$) is feasible with optimal solution \bar{x} and dual solution (\bar{s}, \bar{y}) . Then every x with $x_J = x_J^k$ satisfying the constraints $Ax = b$ and*

$$\begin{aligned} -\|\bar{x}_{i1}\|x_{i0} + \bar{x}_{i1}^T x_{i1} &\leq 0, \quad i \in I_a(\bar{x}), \\ -x_{i0} &\leq 0, \quad i \in I_0(\bar{x}), \quad \bar{s}_{i0} = 0, \\ -x_{i0} - \frac{1}{\bar{s}_{i0}} \bar{s}_{i1}^T x_{i1} &\leq 0, \quad i \in I_0(\bar{x}), \quad \bar{s}_{i0} > 0, \quad \bar{x} \in T, \end{aligned} \quad (5.1)$$

where I_a and I_0 are defined by (3.1), satisfies $c^T x \geq c^T \bar{x}$.

Proof. Assume x , with $x_J = \bar{x}_J$ satisfies $Ax = b$ and (5.1), namely

$$(\nabla g_i(\bar{x}))_J^T (x_{\bar{J}} - \bar{x}_{\bar{J}}) \leq 0, \quad i \in I_a(\bar{x}), \quad (5.2)$$

$$(\bar{\xi}_i)_J^T (x_{\bar{J}} - \bar{x}_{\bar{J}}) \leq 0, \quad i \in I_0(\bar{x}), \quad (5.3)$$

$$A(x - \bar{x}) = 0, \quad (5.4)$$

with $\bar{\xi}_i$ from Lemma 3.1 and where the last equation follows from $A\bar{x} = b$. Due to A2 we know that there exist $\mu \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}_+^{|I_0 \cup I_a|}$ satisfying the KKT conditions (3.2) of ($NLP(x_J^k)$) in \bar{x} , that is

$$\begin{aligned} -c_i &= A_i^T \mu + \lambda_i \bar{\xi}_i, & i \in I_0(\bar{x}), \\ -c_i &= A_i^T \mu + \lambda_i \nabla g_i(\bar{x}), & i \in I_a(\bar{x}), \\ -c_i &= A_i^T \mu, & i \notin I_0(\bar{x}) \cup I_a(\bar{x}) \end{aligned} \quad (5.5)$$

with the subgradients $\bar{\xi}_i$ chosen from Lemma 3.1. Farkas' Lemma (cf. [20]) states that (5.5) is equivalent to the fact that as long as $(x - \bar{x})$ satisfies (5.2) - (5.4), then $c_{\bar{J}}^T (x_{\bar{J}} - \bar{x}_{\bar{J}}) \geq 0 \Leftrightarrow c_{\bar{J}}^T x_{\bar{J}} \geq c_{\bar{J}}^T \bar{x}_{\bar{J}}$ must hold. \square

In the case that ($NLP(x_J^k)$) is infeasible, we can show that the subgradients (4.6) of Lemma 4.1 together with the gradients of the differentiable functions g_i in the solution of ($F(x_J^k)$) provide inequalities that separate the last integer solution.

LEMMA 5.2. *Assume A1 and A2 hold. If ($NLP(x_J^k)$) is infeasible and thus (\bar{x}, \bar{u}) solves ($F(x_J^k)$) with positive optimal value $\bar{u} > 0$, then every x satisfying the linear equalities $Ax = b$ with $x_J = x_J^k$, is infeasible in the constraints*

$$\begin{aligned} -x_{i0} + \frac{\bar{x}_{i1}^T}{\|\bar{x}_{i1}\|} x_{i1} &\leq 0, \quad i \in I_{F1}(\bar{x}), \\ -x_{i0} - \frac{\bar{s}_{i1}^T}{\bar{s}_{i0}} x_{i1} &\leq 0, \quad i \in I_{F0}, \bar{s}_{i0} \neq 0, \\ -x_{i0} &\leq 0, \quad i \in I_{F0}, \bar{s}_{i0} = 0, \end{aligned} \quad (5.6)$$

where I_{F1} and I_{F0} are defined by (4.1) and (\bar{s}, \bar{y}) is the solution of the dual program $(F(x_J^k)\text{-}D)$ of $(F(x_J^k))$.

Proof. The proof is done in analogy to Lemma 1 in [1]. Due to assumption A1 and A2, the optimal solution of $(F(x_J^k))$ is attained. We further know from Lemma 4.1, that there exist $\lambda_{g_i} \geq 0$, with $\sum_{i \in I_F} \lambda_{g_i} = 1$, μ_A and μ_J satisfying the KKT conditions

$$\sum_{i \in I_{F1}} \nabla g_i(\bar{x}) \lambda_{g_i} + \sum_{i \in I_{F0}} \xi_i^n \lambda_{g_i} + A^T \mu_A + I_J^T \mu_J = 0 \quad (5.7)$$

in \bar{x} with subgradients (4.6). To show the result of the lemma, we assume now that x , with $x_J = x_J^k$, satisfies $Ax = b$ and conditions (5.6) which are equivalent to

$$\begin{aligned} g_i(\bar{x}) + \nabla g_i(\bar{x})^T (x - \bar{x}) &\leq 0, \quad i \in I_{F1}(\bar{x}), \\ g_i(\bar{x}) + \xi_i^{n,T} (x - \bar{x}) &\leq 0, \quad i \in I_{F0}(\bar{x}). \end{aligned}$$

We multiply the inequalities by $(\lambda_g)_i \geq 0$ and add all inequalities. Since $g_i(\bar{x}) = \bar{u}$ for $i \in I_F$ and $\sum_{i \in I_F} \lambda_{g_i} = 1$ we get

$$\begin{aligned} \sum_{i \in I_{F1}} (\lambda_{g_i} \bar{u} + \lambda_{g_i} \nabla g_i(\bar{x})^T (x - \bar{x})) + \sum_{i \in I_{F0}} (\lambda_{g_i} \bar{u} + \lambda_{g_i} \xi_i^{n,T} (x - \bar{x})) &\leq 0 \\ \Leftrightarrow \quad \bar{u} + \left(\sum_{i \in I_{F1}} \lambda_{g_i} \nabla g_i(\bar{x}) + \sum_{i \in I_{F0}} (\lambda_{g_i} \xi_i^n) \right)^T (x - \bar{x}) &\leq 0. \end{aligned}$$

Insertion of (5.7) yields

$$\begin{aligned} \bar{u} + (-A^T \mu_A - I_J^T \mu_J)^T (x - \bar{x}) &\leq 0 \\ \Leftrightarrow^{Ax=A\bar{x}=b} \bar{u} - \mu_J^T (x_J - \bar{x}_J) &\leq 0 \\ \Leftrightarrow^{x_J=x_J^k=\bar{x}_J} \bar{u} &\leq 0. \end{aligned}$$

This is a contradiction to the assumption $\bar{u} > 0$. \square

Thus, the solution \bar{x} of $(F(x_J^k))$ produces new constraints (5.6) that strengthen the outer approximation such that the integer solution x_J^k is no longer feasible. If $(NLP(x_J^k))$ is infeasible, the active set $I_F(\bar{x})$ is not empty and thus, at least one constraint (5.6) can be added.

THEOREM 5.1. *Assume A1 and A2. Then Algorithm 1 terminates in a finite number of steps at an optimal solution of (1.1) or with the indication, that it is infeasible.*

Proof. We show that no integer assignment x_J^k is generated twice by showing that $x_J = x_J^k$ is infeasible in the linearized constraints created in

the solutions of $(NLP(x_J^k))$ or $(F(x_J^k))$. The finiteness follows then from the boundedness of the feasible set. A1 and A2 guarantee the solvability, validity of KKT conditions and primal-dual optimality of the nonlinear subproblems $(NLP(x_J^k))$ and $(F(x_J^k))$. In the case, when $(NLP(x_J^k))$ is feasible with solution \bar{x} , Lemma 5.1 states that every \tilde{x} with $\tilde{x}_J = \hat{x}_J$ must satisfy $c^T \tilde{x} \geq c^T \bar{x}$ and is thus infeasible in the constraint $c^T \tilde{x} < c^T \bar{x}$ included in $(LP^k(T, S))$. In the case, when $(NLP(x_J^k))$ is infeasible, Lemma 5.2 yields the result for $(F(x_J^k))$. \square

Modified algorithm avoiding A2. We now present an adaption of Algorithm 1 which is still convergent if the convergence assumption A2 is not valid for every subproblem. Assume N^k is a node such that A2 is violated by $(NLP(x_J^k))$ and assume x with integer assignment $x_J = x_J^k$ is feasible for the updated outer approximation. Then the inner while-loop in step 2b becomes infinite and Algorithm 1 does not converge. In that case we solve the SOCP relaxation (SOC^k) in node N^k . If that problem is not infeasible and has no integer feasible solution, we branch on the solution of this SOCP relaxation to explore the subtree of N^k . Hence, we substitute step 2b by the following step.

```

2b'. solve  $(LP^k(T, S))$  with solution  $x^k$ , set repeat = true.
while ((( $LP^k(T, S)$ ) feasible) & ( $x_J^k$  integer) & ( $c^T x^k < CUB$ ) & repeat)
    save  $x_J^{old} = x_J^k$ 
    if ( $NLP(x_J^k)$  is feasible with solution  $\bar{x}$ )
         $T := T \cup \{\bar{x}\}$ ,
        if ( $c^T \bar{x} < CUB$ )  $CUB = c^T \bar{x}$ ,  $x^* = \bar{x}$  endif
        else compute solution  $\bar{x}$  of  $F(x_J^k)$ ,  $S := S \cup \{\bar{x}\}$ 
        endif
        compute solution  $x^k$  of updated  $(LP^k(T, S))$ 
        if ( $x_J^{old} == x_J^k$ ) set repeat = false endif
    endwhile
    if (!repeat)
        solve nonlinear relaxation  $(SOC^k)$  at the node  $N^k$  with solution  $\bar{x}$ 
         $T := T \cup \{\bar{x}\}$ 
        if ( $\bar{x}_J$  integer): if  $c^T \bar{x} < CUB$ :  $CUB = c^T \bar{x}$ ,  $x^* = \bar{x}$  endif
        go to 2.
        else set  $x^k = \bar{x}$ .
        endif
    endif

```

Note that every subgradient of a conic constraint provides a valid linear outer approximation of the form (2.1). Thus, in the case that we cannot identify the subgradients satisfying the KKT system of $(NLP(x_J^k))$, we take an arbitrary subgradient to update the linear outer approximation $(LP^k(T, S))$.

LEMMA 5.3. *Assume A1 holds. Then Algorithm 2, which is Algorithm 1, where Step 2b is replaced by 2b', terminates in a finite number of steps at an optimal solution of (1.1) or with the indication that it is infeasible.*

Proof. If A2 is not satisfied, we have no guarantee that the linearization in the solution \bar{x} of ($NLP(x_j^k)$) separates the current integer solution. Hence, assume the solution of ($LP^k(T, S)$) is integer feasible with solution x^k and the same integer assignment x_j^k is optimal for the updated outer approximation ($LP^k(T \cup \{\bar{x}\}, S)$) or ($LP^k(T, S \cup \{\bar{x}\})$). Then the nonlinear relaxation (SOC^k) is solved at the current node N^k . If the problem (SOC^k) is not infeasible and its solution is not integer, the algorithm branches on its solution producing new nodes N^{i+1} and N^{i+2} . These nodes are again searched using Algorithm 1 as long as the situation of repeated integer solutions does not occur. Otherwise it is again branched on the solution of the continuous relaxation. If this is done for a whole subtree, the algorithm coincides with a nonlinear branch-and-bound search for this subtree which is finite due to the boundedness of the integer variables. \square

REMARKS. The convergence result of Theorem 5.1 can be directly extended to any outer approximation approach for (1.1) which is based on the properties of ($MIP(T, S)$) (and thus ($LP^k(T, S)$)) proved in Lemma 5.1 and Lemma 5.2. In particular, convergence of the classical outer approximation approach as well as the Generalized Benders Decomposition approach (cf. [6], [4] or [27]) is naturally implied.

Furthermore, our convergence result can be generalized to arbitrary mixed integer programming problems with subdifferentiable convex constraint functions, if it is possible to identify the subgradients that satisfy the KKT system in the solutions of the associated nonlinear subproblems.

6. Numerical experiments. We implemented the modified version of the outer approximation approach Algorithm 2 ('B&B-OA') as well as a nonlinear branch-and-bound approach ('B&B'). The SOCP problems are solved with our own implementation of an infeasible primal-dual interior point approach (cf. [26], Chapter 1), the linear programs are solved with CPLEX 10.0.1.

We report results for mixed 0-1 formulations of different ESTP test problems (instances t₋₄*, t₋₅*) from Beasley's website [21] (cf. [16]) and some problems arising in the context of turbine balancing (instances Test*). The data sets are available on the web [29]. Each instance was solved using the nonlinear branch-and-bound algorithm as well as Algorithm 2, once for the choice $L = 10$ and for the choice $L = 10000$ respectively.

We used best first node selection and pseudocost branching in the nonlinear branch-and-bound approach and depth first search as well as most fractional branching in Algorithm 2, since those performed best in former tests.

Table 1 gives an overview of the problem dimensions according to the notation in this paper. We also list for each problem the number of

TABLE 1

Problem sizes ($m, n, noc, |J|$) and maximal constraints of LP approximation (m_{oa}).

Problem	n	m	noc	$ J $	m_{oa} (L=10)	m_{oa} (L=10000)
t4_nr22	67	50	49	9	122	122
t4_nrA	67	50	49	9	213	231
t4_nrB	67	50	49	9	222	240
t4_nrC	67	50	49	9	281	272
t5_nr1	132	97	96	18	1620	1032
t5_nr21	132	97	96	18	2273	1677
t5_nrA	132	97	96	18	1698	998
t5_nrB	132	97	96	18	1717	1243
t5_nrC	132	97	96	18	1471	1104
Test07	84	64	26	11	243	243
Test07_an	84	63	33	11	170	170
Test54	366	346	120	11	785	785
Test07GF	87	75	37	12	126	110
Test54GF	369	357	131	12	1362	1362
Test07_lowb	212	145	153	56	7005	2331
Test07_lowb_an	210	145	160	56	1730	308

TABLE 2
Number of solved SOCP/LP problems.

Problem	B&B (SOCP)	B&B-OA (L=10) (SOCP/LP)	B&B-OA (L=10000) (SOCP/LP)
t4_nr22	31	9/15	9/15
t4_nrA	31	19/39	20/40
t4_nrB	31	20/40	21/41
t4_nrC	31	26/43	25/43
t5_nr1	465	120/745	52/720
t5_nr21	613	170/ 957	88/1010
t5_nrA	565	140/ 941	50/995
t5_nrB	395	105 / 519	64/552
t5_nrC	625	115/761	56 / 755
Test07	13	8/20	8/20
Test07_an	7	5/9	5/9
Test54	7	5/9	5/9
Test07GF	41	5/39	2/35
Test54GF	37	11/68	9/63
Test07_lowb	383	392/3065	115/2599
Test07_lowb_an	1127	128/ 1505	9/1572

TABLE 3
Run times in seconds.

Problem	B&B	B&B-OA (L=10)	B&B-OA (L=10000)
t4_nr22	2.83	0.57	0.63
t4_nrA	2.86	1.33	1.44
t4_nrB	2.46	1.72	1.71
t4_nrC	2.97	1.54	2.03
t5_nr1	128.96	42.22	29.58
t5_nr21	139.86	61.88	20.07
t5_nrA	128.37	53.44	17.04
t5_nrB	77.03	36.55	18.94
t5_nrC	150.79	44.57	16.11
Test07	0.42	0.28	0.26
Test07_an	0.11	0.16	0.14
Test54	4.4	1.33	1.40
Test07GF	2.26	0.41	0.24
Test54GF	32.37	6.95	5.53
Test07_lowb	244.52	499.47	134.13
Test07_lowb_an	893.7	128.44	14.79

constraints of the largest LP relaxation solved during Algorithm 2 ('m_oa'). As depicted in Algorithm 2, every time a nonlinear subproblem is solved, the number of constraints of the outer approximation problem grows by the number of conic constraints active in the solution of that nonlinear subproblem. Thus, the largest fraction of linear programs solved during the algorithm have significantly fewer constraints than ('m_oa').

For each algorithm Table 2 displays the number of solved SOCP nodes and LP nodes whereas Table 3 displays the run times. A comparison of the branch-and-bound approach and Algorithm 2 on the basis of Table 2 shows, that the latter algorithm solves remarkable fewer SOCP problems. Table 3 displays that for almost all test instances, the branch-and-bound based outer approximation approach is preferable regarding running times, since the LP problems stay moderately in size.

For $L = 10$, at every 10th node an additional SOCP problem is solved whereas for $L = 10000$, for our test set no additional SOCP relaxations are solved. In comparison with $L = 10000$, for $L = 10$ more (11 out of 16 instances) or equally many (3 out of 16 instances) SOCP problems are solved, whereas the number of solved LP problems is decreased only for 6 out of 16 instances. Moreover, the number of LPs spared by the additional SOCP solves for $L = 10$ is not significant in comparison with $L = 10000$ (compare Table 2) and the sizes of the LPs for $L = 10000$ stay smaller in most cases, since fewer linearizations are added (compare

Table 1). Hence, with regard to running times, the version with $L = 10000$ outperforms $L = 10$ for almost all test instances, compare [Table 3](#). Thus, for the problems considered, Algorithm 2 with $L = 10000$, i.e., without solving additional SOCPs, achieves the best performance in comparison to nonlinear branch-and-bound as well as Algorithm 1 with $L = 10$.

In addition to the above considered instances, we tested some of the classical portfolio optimization instances provided by Vielma et al. [28] using Algorithm 2 with $L = 10000$. For each problem, we report in [Table 4](#) the dimension of the MISOCP formulation, the dimension of the largest relaxation solved by our algorithm and the dimension of the a priori LP relaxation with accuracy 0.01 that was presented in [10]. For a better comparison, we report the number of columns plus the number of linear constraints as it is done in [10]. The dimensions of the largest LP relaxations solved by our approach are significantly smaller than the dimensions of the LP approximations solved by [10]. Furthermore, in the lifted linear programming approach in [10], every LP relaxation solved during the algorithm is of the specified dimension. In our approach most of the solved LPs are much smaller than the reported maximal dimension. In [Table 5](#) we report the run times and number of solved nodes problems for our algorithm (Alg.2). For the sake of completeness we added the average and maximal run times reported in [10] although it is not an appropriate comparison since the algorithms have not been tested on similar machines. Since our implementation of an interior SOCP solver is not as efficient as a commercial solver like CPLEX which is used in [10], a comparison of times is also difficult. But the authors of [10] report that solving their LP relaxations usually takes longer than solving the associated SOCP relaxation. Thus, we can assume that due to the low dimensions of the LPs solved in our approach and the moderate number of SOCPs, our approach is likely to be faster when using a more efficient SOCP solver.

7. Summary. We presented a branch-and-bound based outer approximation approach using subgradient based linearizations. We proved convergence under a constraint qualification that guarantees strong duality of the occurring subproblems and extended the algorithm such that this assumption can be relaxed. We presented numerical experiments for some application problems investigating the performance of the approach in terms of solved linear and second order cone subproblems as well as run times. We also investigated the sizes of the linear approximation problems.

Comparison to a nonlinear branch-and-bound algorithm showed that the outer approximation approach solves almost all problems in significantly shorter running times and that its performance is best when not solving additional SOCP relaxations. In comparison to the outer approximation based approach by Vielma et al. in [10], we observed that the dimensions of our LP relaxations are significantly smaller which makes our approach competitive..

TABLE 4
Dimension ($m+n$) and maximal LP approximation ($m_{\text{oa}} + n$) (portfolio instances).

Problem	$n + m$	$ J $	$n + m_{\text{oa}}$ [Alg.2]	nums+cols [10]
classical_20_0	105	20	137	769
classical_20_3	105	20	129	769
classical_20_4	105	20	184	769
classical_30_0	155	30	306	1169
classical_30_1	155	30	216	1169
classical_30_3	155	30	207	1169
classical_30_4	155	30	155	1169
classical_40_0	205	40	298	1574
classical_40_1	205	40	539	1574
classical_40_3	205	40	418	1574
classical_50_2	255	50	803	1979
classical_50_3	255	50	867	1979

TABLE 5
Run times and node problems (portfolio instances).

Problem	Sec. [Alg.2]	Nodes [Alg.2] (SOCP/LP)	Sec. [10] (average)	Sec. [10] (max)
classical_20_0	2.62	10/75	0.29	1.06
classical_20_3	0.62	2/9	0.29	1.06
classical_20_4	5.70	32/229	0.29	1.06
classical_30_0	52.70	119/2834	1.65	27.00
classical_30_1	16.13	30/688	1.65	27.00
classical_30_3	8.61	20/247	1.65	27.00
classical_30_4	0.41	1/0	1.65	27.00
classical_40_0	46.07	51/1631	14.84	554.52
classical_40_1	361.62	292/15451	14.84	554.52
classical_40_3	138.28	171/5222	14.84	554.52
classical_50_2	779.74	496 / 19285	102.88	1950.81
classical_50_3	1279.61	561/36784	102.88	1950.81

Acknowledgements. We would like to thank the referees for their constructive comments that were very helpful to improve this paper.

REFERENCES

- [1] R. FLETCHER AND S. LEYFFER, *Solving Mixed Integer Nonlinear Programs by Outer Approximation*, in Mathematical Programming, 1994, **66**: 327–349.
- [2] R.A. STUBBS AND S. MEHROTRA, *A branch-and-cut method for 0-1 mixed convex programming* in Mathematical Programming, 1999, **86**: 515–532.

- [3] I. QUESADA AND I.E. GROSMANN, *An LP/NLP based Branch and Bound Algorithm for Convex MINLP Optimization Problems*, in Computers and Chemical Engineering, 1992, **16**(10, 11): 937–947.
- [4] A.M. GEOFFRION, *Generalized Benders Decomposition*, in Journal of Optimization Theory and Applications, 1972, **10**(4): 237–260.
- [5] M.T. ÇEZIK AND G. IYENGAR, *Cuts for Mixed 0-1 Conic Programming*, in Mathematical Programming, Ser. A, 2005, **104**: 179–200.
- [6] M.A. DURAN AND I.E. GROSSMANN, *An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs*, in Mathematical Programming, 1986, **36**: 307–339.
- [7] F. ALIZADEH AND D. GOLDFARB, *Second-Order Cone Programming*, RUTCOR, Rutgers Center for Operations Research, Rutgers University, New Jersey, 2001.
- [8] P. BONAMI, L.T. BIEGLER, A.R. CONN, G. CORNUÉJOLS, I.E. GROSSMANN, C.D. LAIRD, J. LEE, A. LODI, F. MARGOT, N. SAWAYA, AND A. WÄCHTER, *An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs*, IBM Research Division, New York, 2005.
- [9] R.A. STUBBS AND S. MEHROTRA, *Generating Convex Polynomial Inequalities for Mixed 0-1 Programs*, Journal of global optimization, 2002, **24**: 311–332.
- [10] J.P. VIELMA, S. AHMED, AND G.L. NEMHAUSER, *A Lifted Linear Programming Branch-and-Bound Algorithm for Mixed Integer Conic Quadratic Programs*, INFORMS Journal on Computing, 2008, **20**(3): 438–450.
- [11] A. ATAMTÜRK AND V. NARAYANAN, *Lifting for Conic Mixed-Integer Programming*, BCOL Research report 07.04, 2007.
- [12] A. ATAMTÜRK AND V. NARAYANAN, *Cuts for Conic Mixed-Integer Programming*, Mathematical Programming, Ser. A, DOI 10.1007/s10107-008-0239-4, 2007.
- [13] A. BEN-TAL AND A. NEMIROVSKI, *On Polyhedral Approximations of the Second-Order Cone*, in Mathematics of Operations Research, 2001, **26**(2): 193–205.
- [14] E. BALAS, S. CERIA, AND G. CORNUÉJOLS, *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, in Mathematical Programming, 1993, **58**: 295–324.
- [15] M. FAMPA AND N. MACULAN, *A new relaxation in conic form for the Euclidean Steiner Tree Problem in \mathbb{R}^n* , in RAIRO Operations Research, 2001, **35**: 383–394.
- [16] J. SOUKUP AND W.F. CHOW, *Set of test problems for the minimum length connection networks*, in ACM SIGMAP Bulletin, 1973, **15**: 48–51.
- [17] D. BERTSIMAS AND R. SHIODA, *Algorithm for cardinality-constrained quadratic optimization*, in Computational Optimization and Applications, 2007, **91**: 239–269.
- [18] Y. NESTEROV AND A. NEMIROVSKII, *Interior-Point Polynomial Algorithms in Convex Programming*, SIAM Studies in Applied Mathematics, 2001.
- [19] R.T. ROCKAFELLAR, *Convex Analysis*, Princeton University Press, 1970.
- [20] C. GEIGER AND C. KANZOW, *Theorie und Numerik restringierter Optimierungsaufgaben*, Springer Verlag Berlin Heidelberg New York, 2002.
- [21] J.E. BEASLEY, *OR Library: Collection of test data for Euclidean Steiner Tree Problems*, <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/esteininfo.html>.
- [22] P. BELOTTI, P. BONAMI, J.J. FORREST, L. LADANYI, C. LAIRD, J. LEE, F. MARGOT, AND A. WÄCHTER, *BonMin*, <http://www.coin-or.org/BonMin/>.
- [23] R. FLETCHER AND S. LEYFFER, *User Manual of filterSQP*, http://www.mcs.anl.gov/~leyffer/papers/SQP_manual.pdf.
- [24] C. LAIRD AND A. WÄCHTER, *IPOPT*, <https://projects.coin-or.org/Ipopt>.
- [25] K. ABHISHEK, S. LEYFFER, AND J.T. LINDEROTH, *FilMINT: An Outer Approximation-Based Solver for Nonlinear Mixed Integer Programs*, Argonne National Laboratory, Mathematics and Computer Science Division, 2008.
- [26] S. DREWES, *Mixed Integer Second Order Cone Programming*, PhD Thesis, June, 2009.

- [27] P. BONAMI, M. KILINC, AND J. LINDEROTH, *Algorithms and Software for Convex Mixed Integer Nonlinear Programs* 2009.
- [28] J.P. VIELMA, *Portfolio Optimization Instances* <http://www2.isye.gatech.edu/~jvielma/portfolio/>.
- [29] S. DREWES, *MISOCP Test Instances* <https://www3.mathematik.tu-darmstadt.de/index.php?id=491>.

PERSPECTIVE REFORMULATION AND APPLICATIONS

OKTAY GÜNLÜK* AND JEFF LINDEROTH†

Abstract. In this paper we survey recent work on the perspective reformulation approach that generates tight, tractable relaxations for convex mixed integer *nonlinear* programs (MINLP)s. This preprocessing technique is applicable to cases where the MINLP contains binary indicator variables that force continuous decision variables to take the value 0, or to belong to a convex set. We derive from first principles the perspective reformulation, and we discuss a variety of practical MINLPs whose relaxation can be strengthened via the perspective reformulation. The survey concludes with comments and computations comparing various algorithmic techniques for solving perspective reformulations.

Key words. Mixed-integer nonlinear programming, perspective functions.

AMS(MOS) subject classifications. 90C11, 90C30.

1. Introduction. Over the past two decades, tremendous advances have been made in the ability to solve mixed integer linear programs (MILP)s. A fundamental reason for the vast improvement is the ability to build tight, tractable relaxations of MILPs. The relaxations are built either via problem reformulation (automatically during a preprocessing phase), or dynamically through the addition of cutting planes. In this paper we survey a collection of techniques for obtaining tight relaxations to (convex) Mixed Integer Nonlinear Programs (MINLP)s. We call these preprocessing techniques the *perspective reformulation*, since they rely on replacing the original convex function in the formulation with its so-called *perspective*.

1.1. Motivation. Consider a 0-1 MINLP, of the form

$$\min_{(x,z) \in \mathcal{F}} c(x, z) \quad (1.1)$$

where $\mathcal{F} = R \cap (\mathbb{R}_+^{n-p} \times \mathbb{B}^p)$, \mathbb{B} denotes $\{0, 1\}$, and

$$R \stackrel{\text{def}}{=} \{(x, z) \in \mathbb{R}_+^{n-p} \times [0, 1]^p \mid f_j(x, z) \leq 0 \ \forall j = 1, \dots, m\}.$$

We call the set R a *continuous relaxation* of \mathcal{F} , and we emphasize that R is not unique in the sense that \mathcal{F} can have many different continuous relaxations. Throughout, we will be interested in sets R that are convex. Even under the convexity requirement, the set R is not unique, and any convex

*Mathematical Sciences Department, IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA (gunluk@us.ibm.com).

†Department of Industrial and Systems Engineering, University of Wisconsin-Madison, 1513 University Avenue, Madison, WI 53706, USA (linderoth@wisc.edu). The second author was supported by the US Department of Energy under grants DE-FE02-08ER25861 and DE-FG02-09ER25869, and the National Science Foundation under grant CCF-0830153.

set R' with the property that $R' \cap \mathbb{R}_+^{n-p} \times \mathbb{B}^p = \mathcal{F}$ is a valid continuous relaxation of \mathcal{F} . If we let $\text{conv}(\mathcal{F})$ to denote the convex hull of \mathcal{F} , clearly all continuous relaxations of \mathcal{F} that are convex have to contain $\text{conv}(\mathcal{F})$ and therefore $\text{conv}(\mathcal{F})$ is the smallest convex continuous relaxation of \mathcal{F} .

In the field of MILP, significant effort is spent on obtaining tight relaxations of the feasible set of solutions. This effort is justified by the fact that the optimization problem simply becomes a linear programming (LP) problem if $\text{conv}(\mathcal{F})$ can be explicitly described with linear inequalities. Notice that as the objective function is linear, it is easy to find an optimal solution that is an extreme point of $\text{conv}(\mathcal{F})$, which is guaranteed to be in \mathcal{F} as all extreme points of $\text{conv}(\mathcal{F})$ are in \mathcal{F} .

In MINLP, on the other hand, the optimal solution to the relaxation may occur at a point interior to $\text{conv}(\mathcal{F})$ and as such, it is not guaranteed to be integral. It is, however, easy to transform any problem to one with a linear objective function by moving the nonlinear objective function into the constraints. Specifically, the problem (1.1) can be equivalently stated as

$$\min\{\eta \mid \eta \in \mathbb{R}, (x, z) \in \mathcal{F}, \eta \geq c(x, z)\}. \quad (1.2)$$

We can, therefore, without loss of generality assume that the objective function of (1.1) is linear. Notice that, under this assumption, it is possible to solve the MINLP as a convex nonlinear programming (NLP) problem if $\text{conv}(\mathcal{F})$ can be explicitly described using convex functions. In general, an explicit description of $\text{conv}(\mathcal{F})$ is hard to produce and unlike the linear case, is not necessarily unique.

In this paper, we review the perspective reformulation approach that, given a MINLP with an associated continuous relaxation R (perhaps after applying the transformation (1.2)), produces a smaller (tighter) continuous relaxation R' that contains $\text{conv}(\mathcal{F})$. The advantage of having tight relaxations is that as they approximate \mathcal{F} better, they give better lower bounds, and they are more effective in obtaining optimal integral solutions via an enumeration algorithm.

1.2. The importance of formulation. To emphasize the importance of a tight relaxation, consider the well-known uncapacitated facility location problem (UFLP). Modeling the UFLP as a MILP is a canonical example taught to nearly all integer programming students to demonstrate the impact of a “good” versus “poor” formulation. In the UFLP, each customer in a set J must have his demand met from some facilities in a set I . A binary variable z_i indicates if the facility i is open, and the continuous variable x_{ij} represents the percentage of customer j ’s demands met from facility i .

The logical relationships that a customer may only be served from an open facility may be written algebraically in “aggregated” form

$$\sum_{j \in J} x_{ij} \leq |J|z_i \quad \forall i \in I$$

or in “disaggregated” form

$$x_{ij} \leq z_i \quad \forall i \in I, j \in J. \quad (1.3)$$

Writing the constraints in disaggregated form (1.3) makes a significant difference in the computational performance of MILP solvers. For example, in 2005, Leyffer and Linderoth [21] experiment with a simple branch and bound based MILP solver and report that on the average it took the solver 10,000 times longer when the aggregated formulation is used. For modern commercial MILP solvers, however, both formulations solve almost simultaneously. This is because modern MILP software *automatically* reformulates the aggregated (weak) formulation into the disaggregated (strong) one. We strongly believe that similar performance improvements can be obtained by MINLP solvers by performing automatic reformulation techniques specially developed for MINLP problems, and we hope this survey work will spur this line of research.

A common reformulation technique used by MILP solvers is to recognize simple structures that appear in the formulation and replace them with tight relaxations for these structures. The tightening of these structures leads to the tightening of the overall relaxation. We will follow this same approach here to derive tighter (perspective) relaxations through the study and characterization of convex hulls of simple sets.

1.3. The perspective reformulation. We are particularly interested in simple sets related to “on-off” type decisions. To that end, let z be a binary indicator variable that controls continuous variables x . The perspective reformulation is based on strengthening the natural continuous relaxation of the following “on-off” set:

$$S \stackrel{\text{def}}{=} \left\{ (x, z) \in \mathbb{R}^n \times \mathbb{B} \mid \begin{array}{ll} x = \hat{x} & \text{if } z = 0 \\ x \in \Gamma & \text{if } z = 1 \end{array} \right\},$$

where \hat{x} is a given point (ex: $\hat{x} = 0$), and

$$\Gamma \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n \mid f_j(x) \leq 0 \text{ } j = 1, \dots, m, u \geq x \geq \ell\}$$

is a bounded convex set. (Note that Γ can be convex even when some of the functions f_j defining it are non-convex.)

In this paper we study the convex hull description of sets closely related to S . We present a number of examples where these simple sets appear as substructures, and we demonstrate that utilizing the convex hull description of these sets helps solve the optimization problem efficiently. Closely related to this work is the effort of Frangioni and Gentile [11, 13], who derive a class of cutting planes that significantly strengthen the formulation for MINLPs containing “on-off” type decisions with convex, separable, objective functions, and demonstrate that these inequalities are quite useful in practice. The connection is detailed more in Section 5.3.

The remainder of the paper is divided into 6 sections. Section 2 gives a review of perspective functions and how they can be used to obtain strong MINLP formulations. Section 3 first derives the convex hull description of two simple sets and then uses them as building blocks to describe the convex hull of more complicated sets. Section 4 describes a number of applications where the perspective reformulation technique can be successfully applied. Section 5 contains some discussion about computational approaches for solving relaxations arising from the perspective reformulation. Section 6 demonstrates numerically the impact of perspective reformulation approach on two applications. Concluding remarks are offered in Section 7.

2. Perspective functions and convex hulls. The *perspective* of a given function of $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the function $\tilde{f} : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ defined as follows:

$$\tilde{f}(\lambda, x) = \begin{cases} \lambda f(x/\lambda) & \text{if } \lambda > 0 \\ 0 & \text{if } \lambda = 0 \\ \infty & \text{otherwise.} \end{cases} \quad (2.1)$$

An important property of perspective functions is that \tilde{f} is convex provided that f is convex. A starting point for the use of the perspective function for strong formulations of MINLPs is the work of Ceria and Soares [8].

2.1. Using perspective functions to obtain convex hulls. Ceria and Soares characterize the closure of the convex hull of the union of convex sets using the perspective transformation. The main result of Ceria and Soares is stated (in a simplified form) in Theorem 2.1.

THEOREM 2.1 (Ceria and Soares [8]). *For $t \in T$, let $G^t : \mathbb{R}^n \rightarrow \mathbb{R}^{m_t}$ be a vector-valued function with the property that the corresponding sets*

$$K^t = \{x \in \mathbb{R}^n : G^t(x) \leq 0\}$$

are convex and bounded. Let $K = \text{conv}(\cup_{t \in T} K^t)$. Then $x \in K$ if and only if the following (nonlinear) system is feasible:

$$x = \sum_{t \in T} x^t; \quad \sum_{t \in T} \lambda_t = 1; \quad \tilde{G}^t(\lambda_t, x^t) \leq 0, \quad \lambda_t \geq 0, \quad \forall t \in T. \quad (2.2)$$

Theorem 2.1 provides an extended formulation that describes the set K in a higher dimensional space. The work extends a well-known result from Balas in the case that all the K^t are polyhedral [2]. Also note that G^t being a convex function is sufficient, but not necessary for K^t to be convex.

A similar argument using the perspective functions was used by Stubbs and Mehrotra to formulate a convex programming problem to generate disjunctive cutting planes [25]. Later, Grossmann and Lee apply these same concepts to more general logic-constrained optimization problems known as generalized disjunctive programs [15].

2.2. Computational challenges. There are a number of challenges in using the convex hull characterization of Theorem 2.1 in computation. One challenge is determining an appropriate disjunction such that $\mathcal{F} \subseteq \cup_{t \in T} K^t$. For example, using the disjunction associated with requiring a collection of the variables to be in $\{0, 1\}$ requires $|T|$ to be exponential in the number of variables chosen. In this case, a cutting plane algorithm, like the one suggested by Stubbs and Mehrotra may be appropriate [25].

A second challenge occurs when K^t is not bounded. In this case, the convex hull characterization is more complicated, and a closed form solution may not be known. Ceria and Soares address these complications and suggest a log-barrier approach to its solution [8].

A third challenge arises from the form of the perspective function. By definition, there is a point of nondifferentiability at $\lambda_t = 0$. This may cause difficulty for solvers used to solve the relaxation. Grossmann and Lee suggest to use the perturbed perspective inequality

$$(\lambda + \epsilon)f(x/(\lambda + \epsilon)) \leq 0$$

for a small constant $\epsilon > 0$, which is valid if $f(0) \leq 0$. An improved perturbed expression is suggested by Furman, Sawaya, and Grossmann [14]:

$$((1 - \epsilon)\lambda + \epsilon)f(x/((1 - \epsilon)\lambda + \epsilon)) \leq \epsilon f(0)(1 - \lambda). \quad (2.3)$$

Notice that both expressions give an exact approximation of the perspective inequality as $\epsilon \rightarrow 0$. In addition, inequality (2.3) has the very useful property that it gives the perspective inequality at $\lambda = 0$ and $\lambda = 1$, for any value of $0 < \epsilon < 1$. Furthermore, it preserves convexity as the left hand side of the inequality is convex if f is convex, and the inequality always forms a relaxation for any choice of $0 < \epsilon < 1$.

When the sets K^t are defined by conic quadratic inequalities (CQI), it is easier to deal with the nondifferentiability issue as the perspective of the associated functions are known to be representable by CQI [3]. We discuss this further in Section 5.2 and give CQI representation of different perspective functions that arise in the MINLP applications considered in Section 4.

3. Simple sets. We next apply Theorem 2.1 to the special case where $T = 2$, and the sets K^0 and K^1 have a specific, simple, structure. More precisely, we consider the cases when K^0 is either a single point or a ray and K^1 is defined by convex functions. We then use these sets as building blocks to describe the convex hull of more complicated sets which appear as sub-structures in some MINLP models. We also note that there is ongoing work on other special cases by Bonami, Cornuéjols, and Hijazi [5].

3.1. The convex hull of a point and a convex set. Consider the set $W = W^0 \cup W^1$ which is defined using the indicator variable $z \in \{0, 1\}$ as follows:

$$W^0 = \{(x, z) \in \mathbb{R}^{n+1} : x = 0, z = 0\}$$

and

$$W^1 = \{(x, z) \in \mathbb{R}^{n+1} : f_i(x) \leq 0 \text{ for } i \in I, u \geq x \geq l, z = 1\}$$

where $u, l \in \mathbb{R}_+$, and I is the index set for the constraints. Clearly, both W^0 and W^1 are bounded, and W^0 is a convex set. Furthermore, if W^1 is also convex then we may write an extended formulation as

$$\begin{aligned} \text{conv}(W) = & \left\{ (x, z) \in \mathbb{R}^{n+1} : \begin{array}{l} 1 \geq \lambda \geq 0, \\ x = \lambda x^1 + (1 - \lambda)x^0, \\ z = \lambda z^1 + (1 - \lambda)z^0, \\ x^0 = 0, z^0 = 0, z^1 = 1 \\ f_i(x^1) \leq 0 \text{ for } i \in I, u \geq x^1 \geq l \end{array} \right\}. \end{aligned}$$

We next give a description of W without the additional variables.

LEMMA 3.1. *If W^1 is convex, then $\text{conv}(W) = W^- \cup W^0$, where*

$$W^- = \left\{ (x, z) \in \mathbb{R}^{n+1} : f_i(x/z) \leq 0 \text{ for } i \in I, uz \geq x \geq lz, 1 \geq z > 0 \right\}$$

(notice that z is strictly positive).

Proof. As x^0, z^0 and z^1 are fixed in the extended formulation above, it is possible to substitute out these variables. In addition, as $z = \lambda$ after these substitutions, we can eliminate λ . Furthermore, as $x = \lambda x^1 = zx^1$, we can eliminate x^1 by replacing it with x/z provided that $z > 0$. If, on the other hand, $z = 0$, clearly $(x, 0) \in \text{conv}(W)$ if and only if $(x, 0) \in W^0$. \square

It is also possible to show that W^0 is contained in the closure of W^- (see [17]) which leads to the following observation.

COROLLARY 3.1. $\text{conv}(W) = \text{closure}(W^-)$.

We would like to emphasize that even when $f(x)$ is a convex function $f_i(x/z)$ may not be convex. However, for $z > 0$ we have

$$f_i(x/z) \leq 0 \Leftrightarrow z^q f_i(x/z) \leq 0 \quad (3.1)$$

for any $q \in \mathbb{R}$. In particular, taking $q = 1$ gives the perspective function which is known to be convex provided that $f(x)$ is convex. Consequently, the set W^- described in Lemma 3.1 can also be written as follows:

$$W^- = \left\{ (x, z) \in \mathbb{R}^{n+1} : zf_i(x/z) \leq 0 \text{ for } i \in I, uz \geq x \geq lz, 1 \geq z > 0 \right\}.$$

When all $f_i(x)$ that define W^1 are polynomial functions, the convex hull of W can be described in closed form in the original space of variables.

More precisely, let

$$f_i(x) = \sum_{t=1}^{p_i} c_{it} \prod_{j=1}^n x_j^{q_{itj}}$$

for all $i \in I$ and define $q_{it} = \sum_{j=1}^n q_{itj}$, $q_i = \max_t \{q_{it}\}$ and $\bar{q}_{it} = q_i - q_{it}$. If all $f_i(x)$ are convex and bounded in $[l, u]$, then (see [17])

$$\text{conv}(W) = \left\{ (x, z) \in \mathbb{R}^{n+1} : \begin{array}{l} \sum_{t=1}^{p_i} c_{it} z^{\bar{q}_{it}} \prod_{j=1}^n x_j^{q_{itj}} \leq 0 \text{ for } i \in I, \\ zu \geq x \geq lz, 1 \geq z \geq 0 \end{array} \right\}.$$

3.2. The convex hull of a ray and a convex set. It is possible to extend Lemma 3.1 to obtain the convex hull description of a ray and a convex set that contains the ray as an unbounded direction. More precisely consider the set $T = T_0 \cup T_1$ where

$$T^0 = \{(x, y, z) \in \mathbb{R}^{n+1+1} : x = 0, y \geq 0, z = 0\},$$

and

$$T^1 = \{(x, y, z) \in \mathbb{R}^{n+1+1} : f_i(x) \leq 0 \text{ for } i \in I, g(x) \leq y, u \geq x \geq l, z = 1\}$$

where $u, l \in \mathbb{R}_+^n$, and $I = \{1, \dots, t\}$.

LEMMA 3.2. *If T^1 is convex, then $\text{conv}(T) = T^- \cup T^0$, where*

$$T^- = \left\{ (x, y, z) \in \mathbb{R}^{n+1+1} : \begin{array}{l} f_i(x/z) \leq 0 \text{ for } i \in I, g(x/z) \leq y/z, \\ uz \geq x \geq lz, 1 \geq z > 0 \end{array} \right\}.$$

Proof. Using the same arguments as in the proof of Lemma 3.1, it is easy to show that $\text{conv}(T) = P \cup T^0$, where the following gives an extended formulation for the set P :

$$P = \left\{ (x, y, z) \in \mathbb{R}^{n+1+1} : \begin{array}{l} f_i(x/z) \leq 0 \text{ for } i \in I, uz \geq x \geq lz, 1 \geq z > 0 \\ g(x/z) \leq y/z - \frac{1-z}{z} y^0, y^0 \geq 0 \end{array} \right\}.$$

As $(1-z)/z > 0$ and $y^0 \geq 0$ for all feasible points, y^0 can easily be projected out to show that $P = T^-$. \square

Similar to the proof of Corollary 3.1, it is possible to show that T^0 is contained in the closure of T^- .

COROLLARY 3.2. $\text{conv}(T) = \text{closure}(T^-)$.

In addition, we note that when all $f_i(x)$ for $i \in I$ and $g(x)$ are polynomial functions, the convex hull of T can be described in closed form by simply multiplying each inequality in the description of T^- with z raised to an appropriate power. We do not present this description to avoid repetition.

3.3. A simple quadratic set. Consider the following mixed-integer set with 3 variables:

$$S = \left\{ (x, y, z) \in \mathbb{R}^2 \times \mathbb{B} : y \geq x^2, uz \geq x \geq lz, x \geq 0 \right\}.$$

Notice that $S = S^0 \cup S^1$ where $S^0 = \{(0, y, 0) \in \mathbb{R}^3 : y \geq 0\}$, and

$$S^1 = \{(x, y, 1) \in \mathbb{R}^3 : y \geq x^2, u \geq x \geq l, x \geq 0\}.$$

Applying Lemma 3.2 gives the convex hull of S as the perspective of the quadratic function defining the set. Note that when $z > 0$ the constraint $yz \geq x^2$ is same as $y/z \geq (x/z)^2$ and when $z = 0$, it implies that $x = 0$.

LEMMA 3.3. $\text{conv}(S) = S^c$ where

$$S^c = \{(x, y, z) \in \mathbb{R}^3 : yz \geq x^2, uz \geq x \geq lz, 1 \geq z \geq 0, x, y \geq 0\}.$$

Notice that $x^2 - yz$ is not a convex function and yet the set $T^c = \{(x, y, z) \in \mathbb{R}^3 : yz \geq x^2, x, y, z \geq 0\}$ is a convex set. This explains why the set S^c , obtained by intersecting T^c with half-spaces, is convex.

3.4. A larger quadratic set. Using the convex hull description of the set S , it is possible to produce a convex hull description of the following set

$$Q = \left\{ (w, x, z) \in \mathbb{R}^{n+1} \times \mathbb{B}^n : w \geq \sum_{i=1}^n q_i x_i^2, u_i z_i \geq x_i \geq l_i z_i, i \in I \right\}, \quad (3.2)$$

where $I = \{1, \dots, n\}$ and $q, u, l \in \mathbb{R}_+^n$. The convex hull description of Q is closely related to the convex envelope of the function $\sum_{i=1}^n q_i x_i^2$ over a mixed integer set. This set was first considered in the Ph.D. thesis of Stubbs [26].

Now consider the following extended formulation of Q

$$\bar{Q} \stackrel{\text{def}}{=} \left\{ (w, x, y, z) \in \mathbb{R}^{3n+1} : w \geq \sum_i q_i y_i, (x_i, y_i, z_i) \in S_i, i \in I \right\}$$

where S_i has the same form as the set S discussed in the previous section except the bounds u and l are replaced with u_i and l_i . Note that if $(w, x, y, z) \in \bar{Q}$ then $(w, x, z) \in Q$, and therefore $\text{proj}_{(w,x,z)}(\bar{Q}) \subseteq Q$. On the other hand, for any $(w, x, z) \in Q$, letting $y'_i = x_i^2$ gives a point $(w, x, y', z) \in \bar{Q}$. Therefore, \bar{Q} is indeed an extended formulation of Q , or, in other words, $Q = \text{proj}_{(w,x,z)}(\bar{Q})$.

Before we present a convex hull description of \bar{Q} we first define some basic properties of mixed-integer sets. First, remember that given a closed set $P \subset \mathbb{R}^n$, a point $p \in P$ is called an *extreme point* of P if it can not be represented as $p = 1/2p_1 + 1/2p_2$ for $p_1, p_2 \in P$, $p_1 \neq p_2$. The set P is called *pointed* if it has extreme points. A pointed set P is called *integral*

with respect to (w.r.t.) a subset of the indices J if for any extreme point $p \in P$, $p_i \in \mathbb{Z}$ for all $i \in J$.

LEMMA 3.4 ([17]). *For $i = 1, 2$ let $P_i \subset \mathbb{R}^{n_i}$ be a closed and pointed set which is integral w.r.t. indices I_i . Let*

$$P' = \{(x, y) \in \mathbb{R}^{n_1+n_2} : x \in P_1, y \in P_2\},$$

then,

(i) P' is integral with respect to $I_1 \cup I_2$.

(ii) $\text{conv}(P') = \{(x, y) \in \mathbb{R}^{n_1+n_2} : x \in \text{conv}(P_1), y \in \text{conv}(P_2)\}$.

LEMMA 3.5 ([17]). *Let $P \subset \mathbb{R}^n$ be a given closed, pointed set and let*

$$P' = \{(w, x) \in \mathbb{R}^{n+1} : w \geq ax, x \in P\}$$

where $a \in \mathbb{R}^n$.

(i) *If P is integral w.r.t. J , then P' is also integral w.r.t. J .*

(ii) $\text{conv}(P') = P''$ where

$$P'' = \{(w, x) \in \mathbb{R}^{n+1} : w \geq ax, x \in \text{conv}(P)\}.$$

We are now ready to present the convex hull of \bar{Q} .

LEMMA 3.6. *The set*

$$\bar{Q}^c = \left\{ (w, x, y, z) \in \mathbb{R}^{3n+1} : w \geq \sum_i q_i y_i, (x_i, y_i, z_i) \in S_i^c, i \in I \right\}.$$

is integral w.r.t. the indices of z variables. Furthermore, $\text{conv}(\bar{Q}) = \bar{Q}^c$.

Proof. Let

$$D = \{(x, y, z) \in \mathbb{R}^{3n} : (x_i, y_i, z_i) \in S_i, i \in I\}$$

so that

$$\bar{Q} = \{(w, x, y, z) \in \mathbb{R}^{3n+1} : w \geq \sum_{i=1}^n q_i y_i, (x, y, z) \in D\}.$$

By Lemma 3.5, the convex hull of \bar{Q} can be obtained by replacing D with its convex hull in this description. By Lemma 3.4, this can simply be done by taking convex hulls of S_i 's, that is, by replacing S_i with $\text{conv}(S_i)$ in the description of D . Finally, by Lemma 3.5, \bar{Q}^c is integral. \square

A natural next step is to study the projection of the set \bar{Q}^c into the space of (w, x, z) . One possibility is to substitute the term x_i^2/z_i for each variable y_i , resulting in the inequality $w \geq \sum_i q_i x_i^2/z_i$. This formula, however, may not be suitable for computation as it is not defined for $z_i = 0$, and $z_i = 0$ is one of the two feasible values for z_i . We next present an

explicit description of the projection that uses an exponential number of inequalities. Let

$$Q^c = \left\{ (w, x, z) \in \mathbb{R}^{2n+1} : w \prod_{i \in S} z_i \geq \sum_{i \in S} q_i x_i^2 \prod_{l \in S \setminus \{i\}} z_l, \forall S \subseteq I \right. \\ \left. u_i z_i \geq x_i \geq l_i z_i, x_i \geq 0, i \in I \right\}. \quad (\text{II})$$

Notice that a given point $\bar{p} = (\bar{w}, \bar{x}, \bar{z})$ satisfies the nonlinear inequalities in the description of Q^c for a particular $S \subseteq I$ if and only if one of the following conditions hold: (i) $\bar{z}_i = 0$ for some $i \in S$, or, (ii) if all $z_i > 0$, then $\bar{w} \geq \sum_{i \in S} q_i \bar{x}_i^2 / \bar{z}_i$. Based on this observation it is possible to show that these (exponentially many) inequalities are sufficient to describe the convex hull of Q in the space of the original variables.

LEMMA 3.7 ([17]). $Q^c = \text{proj}_{(w, x, z)}(\bar{Q}^c)$. Note that all of the exponentially many inequalities that are used in the description of Q^c are indeed necessary. To see this, consider a simple instance with $u_i = l_i = q_i = 1$ for all $i \in I = \{1, 2, \dots, n\}$. For a given $\bar{S} \subseteq I$, let $p^{\bar{S}} = (\bar{w}, \bar{x}, \bar{z})$ where $\bar{w} = |\bar{S}| - 1$, $\bar{z}_i = 1$ if $i \in \bar{S}$, and $\bar{z}_i = 0$ otherwise, and $\bar{x} = \bar{z}$. Note that $p^{\bar{S}} \notin Q^c$. As $\bar{z}_i = q_i \bar{x}_i^2$, inequality (II) is satisfied by \bar{p} for $S \subseteq I$ if and only if

$$(|\bar{S}| - 1) \prod_{i \in S} \bar{z}_i \geq |S| \prod_{i \in S} \bar{z}_i.$$

Note that unless $S \subseteq \bar{S}$, the term $\prod_{i \in S} \bar{z}_i$ becomes zero and therefore inequality (II) is satisfied. In addition, inequality (II) is satisfied whenever $|\bar{S}| > |S|$. Combining these two observations, we can conclude that the only inequality violated by $p^{\bar{S}}$ is the one with $S = \bar{S}$. Due to its size, the projected set is not practical for computational purposes and we conclude that it is more advantageous to work in the extended space, keeping the variables y_i

3.5. A simple non-quadratic set. The simple 3 variable mixed-integer set S introduced in Section 3.3 can be generalized to the following set, studied by Aktürk, Atamtürk, and Gürel [1]:

$$C = \left\{ (x, y, z) \in \mathbb{R}^2 \times \mathbb{B} : y \geq x^{a/b}, uz \geq x \geq lz, x \geq 0 \right\}$$

where $a, b \in \mathbb{Z}_+$ and $a \geq b > 0$. Clearly $C = C^0 \cup C^1$, with

$$C^0 = \{(0, y, 0) \in \mathbb{R}^3 : y \geq 0\},$$

and

$$C^1 = \{(x, y, 1) \in \mathbb{R}^3 : y \geq x^{a/b}, u \geq x \geq l, x \geq 0\}.$$

By applying Lemma 3.2, the convex hull of C is given by using the perspective of the function $f(y, x) = y^b - x^a$ and scaling the resulting inequality by z^a .

LEMMA 3.8 (Aktürk, Atamtürk, Gürel [1]). *The convex hull of C is given by*

$$C^c = \{(x, y, z) \in \mathbb{R}^3 : y^b z^{a-b} \geq x^a, uz \geq x \geq lz, 1 \geq z \geq 0, x, y \geq 0\}.$$

In addition, it is possible to construct the convex hull of larger sets using the set C as a building block. See [1] for more details.

4. Applications. In this section, we present six applications to which the perspective reformulation has been applied.

4.1. Separable Quadratic UFL. The Separable Quadratic Uncapacitated Facility Location Problem (SQUFL) was introduced by Günlük, Lee, and Weismantel [16]. In the SQUFL, there is a set of customers J , opening a facility $i \in I$. All customers have unit demand that can be satisfied using open facilities only. The shipping cost is proportional to the square of the quantity delivered. Letting z_i indicate if facility $i \in I$ is open, and x_{ij} denote the fraction of customer j 's demand met from facility i , SQUFL can be formulated as follows:

$$\begin{aligned} & \min \sum_{i \in I} c_i z_i + \sum_{i \in I} \sum_{j \in J} q_{ij} x_{ij}^2 \\ \text{subject to } & x_{ij} \leq z_i \quad \forall i \in I, \forall j \in J, \\ & \sum_{i \in I} x_{ij} = 1 \quad \forall j \in J, \\ & z_i \in \{0, 1\}, \quad x_{ij} \geq 0 \quad \forall i \in I, \forall j \in J. \end{aligned}$$

To apply the perspective formulation, auxiliary variables y_{ij} are used to replace the terms x_{ij}^2 in the objective function and the constraints

$$x_{ij}^2 - y_{ij} \leq 0 \quad \forall i \in I, j \in J \tag{4.1}$$

are added. In this reformulation, if $z_i = 0$, then $x_{ij} = 0$ and $y_{ij} \geq 0 \forall j \in J$, while if $z_i = 1$, the convex nonlinear constraints (4.1) should also hold. Therefore, we can strengthen the formulation of SQUFL by replacing (4.1) by its perspective reformulation

$$x_{ij}^2/z_i - y_{ij} \leq 0 \quad \forall i \in I, \forall j \in J. \tag{4.2}$$

We demonstrate the impact of this reformulation on solvability of the MINLP in Section 6.1.

4.2. Network design with congestion constraints. The next application is a network design problem with requirements on queuing delay. Similar models appear in the papers [6], [4], and [7]. In the problem, there is a set of commodities K to be shipped over a capacitated directed network $G = (N, A)$. The capacity of arc $(i, j) \in A$ is u_{ij} , and each node $i \in N$ has a net supply b_i^k of commodity $k \in K$. There is a fixed cost c_{ij} of opening each arc $(i, j) \in A$, and we introduce $\{0,1\}$ -variables z_{ij} to indicate whether arc $(i, j) \in A$ is opened. The quantity of commodity k routed on arc (i, j) is measured by variable x_{ij}^k and $f_{ij} = \sum_{k \in K} x_{ij}^k$ denotes the total flow on the arc. A typical measure of the total weighted congestion (or queuing delay) is

$$\rho(f) \stackrel{\text{def}}{=} \sum_{(i,j) \in A} r_{ij} \frac{f_{ij}}{1 - f_{ij}/u_{ij}},$$

where $r_{ij} \geq 0$ is a user-defined weighting parameter for each arc. We use decision variables y_{ij} to measure the contribution of the congestion on arc (i, j) to the total congestion $\rho(f)$. The network should be designed so as to keep the total queuing delay less than a given value β , and this is to be accomplished at minimum cost. The resulting optimization model (NDCC) can be written as

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{ij} z_{ij} \\ \text{subject to} \quad & \sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = b_i^k \quad \forall i \in N, \forall k \in K, \\ & \sum_{k \in K} x_{ij}^k - f_{ij} = 0 \quad \forall (i, j) \in A, \\ & f_{ij} \leq u_{ij} z_{ij} \quad \forall (i, j) \in A, \end{aligned} \quad (4.3)$$

$$y_{ij} \geq \frac{r_{ij} f_{ij}}{1 - f_{ij}/u_{ij}} \quad \forall (i, j) \in A, \quad (4.4)$$

$$\begin{aligned} & \sum_{(i,j) \in A} y_{ij} \leq \beta, \\ x \in \mathbb{R}_+^{|A| \times |K|}, \quad & y \in \mathbb{R}_+^{|A|}, \quad f \in \mathbb{R}_+^{|A|}, \quad z \in \{0, 1\}^{|A|}. \end{aligned}$$

In this formulation of NDCC, note that if $z_{ij} = 0$, then $f_{ij} = 0$ and $y_{ij} \geq 0$. On the other hand, if $z_{ij} = 1$, then f_{ij} and y_{ij} must satisfy $f_{ij} \leq u_{ij}$ and constraint (4.4). Therefore, each constraint (4.4) can be replaced by its perspective counterpart:

$$z_{ij} \left[\frac{r_{ij} f_{ij}/z_{ij}}{1 - f_{ij}/(u_{ij} z_{ij})} - \frac{y_{ij}}{z_{ij}} \right] \leq 0. \quad (4.5)$$

4.3. Scheduling with controllable processing times. Consider a scheduling problem where jobs are assigned to non-identical parallel machines with finite capacity. Let J denote the set of jobs and I denote the set of machines. In this problem, not all jobs have to be processed but if job $j \in J$ is assigned to a machine $i \in I$, a reward of h_{ij} is collected. The regular processing time of job j on machine i is p_{ij} , however by paying a certain cost, it can be reduced to $(p_{ij} - x_{ij})$ where $x_{ij} \in [0, u_{ij}]$. The cost of reducing the processing time of job i on machine j by x_{ij} units is given by the expression

$$f_{ij}(x_{ij}) = k_{ij}x_{ij}^{a_{ij}/b_{ij}}.$$

This problem is called the *machine-job assignment problem with controllable times* and has been recently studied by Aktürk, Atamtürk and Gürel [1]. A MINLP formulation for this problem is:

$$\begin{aligned} \max \quad & \sum_{i \in I} \sum_{j \in J} (h_{ij}z_{ij} - f_{ij}(x_{ij})) \\ \text{subject to} \quad & \sum_{j \in J} (p_{ij}z_{ij} - x_{ij}) \leq c_i \quad \forall i \in I \\ & x_{ij} \leq u_{ij}z_{ij} \quad \forall i \in I, \forall j \in J \quad (4.6) \end{aligned}$$

$$\begin{aligned} & \sum_{i \in I} z_{ij} \leq 1 \quad \forall j \in J \\ & z_{ij} \in \{0, 1\}, \quad x_{ij} \geq 0 \quad \forall i \in I, \forall j \in J \quad (4.7) \end{aligned}$$

where the variable z_{ij} denotes if job j is assigned to machine i and x_{ij} is the reduction on the associated processing time. The total processing time available on machine $i \in I$ is denoted by c_i . The objective is to maximize the sum of the rewards minus the cost of reducing the processing times.

As in the case of the SQUFL in Section 4.1, after adding a new variable y_{ij} and a new constraint

$$x_{ij}^{a_{ij}/b_{ij}} \leq y_{ij} \quad (4.8)$$

for all $i \in I$, $j \in J$, it is possible to replace the objective function with the following linear expression:

$$\sum_{i \in I} \sum_{j \in J} (h_{ij}z_{ij} - y_{ij}/k_{ij}).$$

The inequality (4.8) together with inequalities (4.6) and (4.7) is the set C studied in Section 3.5 and therefore, inequality (4.8) can be replaced with its perspective counterpart

$$z_{ij} \left(\frac{x_{ij}}{z_{ij}} \right)^{a_{ij}/b_{ij}} \leq y_{ij} \quad (4.9)$$

to obtain a stronger formulation. The authors of [1] raise both sides of inequality (4.9) to the b th power and multiply both sides by $z_{ij}^{a_{ij}-b_{ij}}$ to obtain equivalent inequalities

$$x_{ij}^{a_{ij}} \leq y_{ij}^{b_{ij}} z^{a_{ij}-b_{ij}}. \quad (4.10)$$

4.4. The unit commitment problem. One of the essential optimization problems in power generation is the so-called *unit commitment problem* which involves deciding the power output levels of a collection of power generators over a period of time. In this setting, a generator (also called a *unit*) is either turned off and generates no power, or it is turned on and generates power in a given range $[l, u]$. It is important to point out that $l > 0$ and therefore production levels are "semi-continuous". In most models, time horizon is divided into a small number of discrete intervals (ex: 48 half hour intervals for a daily problem) and the generators are required to collectively satisfy a given demand level in each interval. The operating cost of a generator is typically modeled by a convex quadratic function. There are also additional constraints including the most commonly used *min-up*, *min-down* constraints that require that a generator must stay on for a certain number of time periods after it is turned on, and similarly, it must stay down for a number of time periods after it is turned off. Letting I denote the set of generators and T denote the set of time periods under consideration, a MINLP formulation for this problem is the following:

$$\min \sum_{i \in I} \sum_{t \in T} h_{it} z_{it} + \sum_{i \in I} \sum_{t \in T} f_{it}(x_{it})$$

$$\text{subject to } \sum_{i \in I} x_{it} = d_t \quad \forall t \in T \\ l_i z_{it} \leq x_{it} \leq u_i z_{it} \quad \forall i \in I, \forall t \in T \quad (4.11)$$

$$z \in P \quad (4.12)$$

$$z_{ij} \in \{0, 1\} \quad \forall i \in I, \forall t \in T \quad (4.13)$$

where variable z_{it} denotes if generator $i \in I$ is turned on in period $t \in T$ and variable x_{it} gives the production level when the generator is on. There is a fixed cost h_{it} of operating a unit i in period t as well as a variable cost given by the convex quadratic function $f_{it}(x) = a_{it}x^2 + b_{it}x$ for some given $a_{it}, b_{it} \in \mathbb{R}_+$. The demand requirement in period t is given by d_t . Finally, the constraint (4.12) above expresses some other constraints involving how generators can be turned on and off.

To obtain a stronger formulation, it is again possible to introduce new variables y_{it} and constraints

$$a_{it}x_{it}^2 + b_{it}x_{it} \leq y_{it} \quad (4.14)$$

for all $i \in I$ and $t \in T$ so that the new variable y_{it} can replace the term $f_{it}(x_{it})$ in the objective function. Using inequalities (4.11) and (4.13), we can now replace inequality (4.14) with its perspective counterpart

$$a_{it}x_{it}^2 + b_{it}x_{it}z_{it} \leq y_{it}z_{it} \quad (4.15)$$

to obtain a stronger formulation.

4.5. Stochastic service system design. Elhedhli [10] describes a stochastic service system design problem (SSSD) modeled as a network of M/M/1 queues. The instance is characterized by a sets of customers M , facilities N , and service levels K . There are binary decision variables x_{ij} to denote if customer i 's demand is met by facility j and y_{jk} to denote if facility j is operating at service level k . Customer i has a mean demand rate of λ_i and facility j has a mean service rate of μ_{jk} when operated at service level k . There is a fixed cost c_{ij} of assigning customer i to facility j , and a fixed cost f_{jk} of operating facility j at level k .

A straightforward formulation of problem is not convex, however, by introducing auxiliary variables v_j and z_{jk} , Elhedhli provides the following convex MINLP formulation:

$$\begin{aligned} \min \quad & \sum_{i \in M} \sum_{j \in N} c_{ij}x_{ij} + t \sum_{j \in N} v_j + \sum_{j \in N} \sum_{k \in K} f_{jk}y_{jk} \\ \text{subject to} \quad & \sum_{i \in M} \lambda_i x_{ij} - \sum_{k \in K} \mu_{jk} z_{jk} = 0 \quad \forall j \in N \\ & \sum_{j \in N} x_{ij} = 1 \quad \forall i \in M \\ & \sum_{k \in K} y_{jk} \leq 1 \quad \forall j \in N \\ & z_{jk} - y_{jk} \leq 0 \quad \forall j \in N, \forall k \in K \quad (4.16) \\ & z_{jk} - v_j/(1 + v_j) \leq 0 \quad \forall j \in N, \forall k \in K \quad (4.17) \\ & z_{jk}, v_j \geq 0, \quad x_{ij}, y_{jk} \in \{0, 1\} \quad \forall i \in M, j \in N, \forall k \in K \quad (4.18) \end{aligned}$$

Instead of directly including the nonlinear constraints (4.17) in the formulation, Elhedhli proposes linearizing the constraints at points $(v_j, z_{jk}) = (v_j^b, 1)$, $b \in B$, yielding

$$z_{jk} - \frac{1}{(1 + v_j^b)^2} v_j \leq \frac{(v_j^b)^2}{(1 + v_j^b)^2}. \quad (4.19)$$

Elhedhli uses a dynamic cutting plane approach to add inequalities (4.19).

Notice that if $y_{jk} = 0$, then $z_{jk} = 0$ and $v_j \geq 0$, and therefore inequality (4.17) can be replaced by their perspective counterpart

$$z_{jk} \leq \frac{v_j}{1 + v_j/y_{jk}} \quad (4.20)$$

yielding a tighter (non-linear) formulation. Furthermore, linearizing these inequalities at points $(v_j, y_{jk}, z_{jk}) = (v_j^b, 1, 1)$, $b \in B$, gives

$$z_{jk} - \frac{1}{(1 + v_j^b)^2} v_j \leq \frac{(v_j^b)^2}{(1 + v_j^b)^2} y_{jk} \quad (4.21)$$

which dominate the inequalities used in Elhedhli [10]. Note that the inequalities (4.21) could also be derived by applying a logical integer strengthening argument to the inequalities (4.19). The linearized perspective inequalities are called *perspective cuts* [11], which are discussed in greater detail in Section 5.3. Computational results demonstrating the effect of the perspective reformulation on this application are given in Section 6.2.

4.6. Portfolio selection. A canonical optimization problem in financial engineering is to find a minimum variance portfolio that meets a given minimum expected return requirement of $\rho > 0$, see [22]. In the problem, there is a set N of assets available for purchase. The expected return of asset $i \in N$ is given by α_i , and the covariance of the returns between pairs of assets is given in the form of a positive-definite matrix $Q \in \mathbb{R}^{n \times n}$. There can be at most K different assets in the portfolio and there is a minimum and maximum buy-in thresholds for the assets chosen. A MINLP formulation of the problem is

$$\min\{x^T Q x \mid e^T x = 1, \alpha^T x \geq \rho, e^T z \leq K; \ell_i z_i \leq x_i \leq u_i z_i, z_i \in \mathbb{B} \forall i \in N\},$$

where the decision variable x_i is the percentage of the portfolio invested in asset i and z_i is a binary variable indicating the purchase of asset i . Unfortunately, direct application of the perspective reformulation is not possible, as the objective is not a separable function of the decision variables.

However, in many practical applications, the covariance matrix is obtained from a *factor model* and has the form $Q = B\Omega B^T + \Delta^2$, for a given *exposure matrix*, $B \in \mathbb{R}^{n \times f}$, positive-definite *factor-covariance matrix* $\Omega \in \mathbb{R}^{f \times f}$, and positive definite, diagonal *specific-variance* matrix $\Delta \in \mathbb{R}^{n \times n}$ [24]. If a factor model is given, a separable portion of the objective function is easily extracted by introducing variables y_i , changing the objective to

$$\min x^T (B\Omega B^T)x + \sum_{i \in N} \Delta_{ii} y_i,$$

and enforcing the constraints $y_i \geq x_i^2 \forall i \in N$. These constraints can then be replaced by their perspective counterparts $y_i \geq x_i^2/z_i$ to obtain a tighter formulation.

Even if the covariance matrix Q does not directly have an embedded diagonal structure from a factor model, it may still be possible to find a diagonal matrix D such that $R = Q - D$ is positive-definite. For example,

Frangioni and Gentile [11] suggest using $D = \lambda_n I$, where $\lambda_n > 0$ is the smallest eigenvalue of Q . Frangioni and Gentile [12] subsequently gave a semidefinite programming approach to obtain a diagonal matrix D that may have desirable computational properties.

5. Computational approaches. Algorithms to solve MINLP are based on solving a sequence of continuous relaxations of the formulation. In this section, we discuss approaches and software for solving the perspective reformulation. The approaches for solving the perspective reformulation fall into three general categories, with tradeoffs in speed and generality of the approaches. The first approach, for use in the most general cases, is to simply give the reformulated problem to a general purpose NLP solver. The second approach is to use a solver that is specialized for second-order cone programming problems. A final approach is to linearize the nonlinear functions of the perspective reformulation and use an LP solver. This approach is most effective if the linearizations are added in a dynamic manner.

5.1. NLP solvers. Care must be taken when using a traditional solver for nonlinear programs to solve the perspective reformulation. Applying the perspective transformation to a constraint $f(x) \leq 0$ leads to $zf(x/z) \leq 0$, which is not defined when $z = 0$. Often, the constraint $zf(x/z) \leq 0$ can be manipulated to remove z from the denominator, but this may result in new difficulties for the NLP solver. To illustrate this point, consider the inequalities (4.1) in the description of the SQUFL introduced in Section 4.1. Applying the perspective transformation to $x^2 - y \leq 0$ gives

$$x^2/z - y \leq 0, \quad (5.1)$$

which is not defined at $z = 0$. Multiplying both sides of the inequality by z gives

$$x^2 - yz \leq 0. \quad (5.2)$$

However, since the function $x^2 - yz$ is not convex, traditional NLP solvers cannot guarantee convergence to a globally optimal solution to the relaxation. A reformulation trick can be used to turn (5.1) into an equivalent inequality with a convex constraint function. Specifically, since $y, z \geq 0$, (5.1) is equivalent to

$$\sqrt{(2x)^2 + (y-z)^2} - y - z \leq 0, \quad (5.3)$$

and the constraint function in (5.3) is convex. This, however, may introduce yet a different obstacle to NLP software, as the constraint function in (5.3) is not differentiable at $(x, y, z) = (0, 0, 0)$. In Section 6 we will show some computational experiments aimed at demonstrating the effectiveness of NLP software at handling perspective constraints in their various equivalent forms.

5.2. SOCP solvers. A second-order cone program (SOCP) is a mathematical program with (conic) quadratic inequalities of the form

$$\|Ax + b\|_2 \leq c^T x + d. \quad (5.4)$$

A rotated second-order cone constraint is of the form

$$x^2 \leq yz \text{ with } y \geq 0, z \geq 0. \quad (5.5)$$

As noted in Section 5.1, rotated second-order cone constraints (5.5) are equivalent to second-order cone constraints (5.4) since

$$\|(2x, y - z)^T\| \leq y + z \Leftrightarrow x^2 \leq yz, y \geq 0, z \geq 0. \quad (5.6)$$

The set of points that satisfy (5.4) or (5.5) forms a convex set, and efficient and robust algorithms exist for solving optimization problems containing second-order cone constraints [27, 23]. An interesting and important observation from a computational standpoint is that the nonlinear inequalities present in all of the applications described in Section 4 can be described with second order cone constraints. Further, if a set of points is representable using second order cone constraints, then the perspective mapping of the set is also SOC-representable [3]. Therefore, quite often software designed to solve SOCPs can be used to solve the perspective relaxations arising from real applications.

To demonstrate the variety of reformulation techniques required to express nonlinear constraints in their SOC representation, we give the SOC representations for all of the nonlinear perspective constraints appearing in the applications in Section 4. The book of Ben-Tal and Nemirovski [3] contains a wealth of knowledge about the types of inequalities whose feasible regions are representable with CQI.

For the SQUFL described in Section 4.1, the nonlinear inequalities in the perspective reformulation (4.2) can be multiplied by z_i and then are evidently in rotated second order cone form (5.5). The nonlinear inequalities (4.5) in the perspective reformulation of the NDCC described in Section 4.2 can be put in the (rotated) second order cone form

$$(y_{ij} - r_{ij}f_{ij})(u_{ij}z_{ij} - f_{ij}) \geq r_{ij}f_{ij}^2, \quad (5.7)$$

which is a rotated SOC constraint as $y_{ij} \geq r_{ij}f_{ij}$ and $u_{ij}z_{ij} \geq f_{ij}$ for any feasible solution. Note that we multiply the inequality by z_{ij} to obtain the form (5.7) above. For the scheduling application of Section 4.3, the nonlinear inequalities (4.10) can be represented using SOC constraints. In this case, the transformation is more complicated, requiring $O(\log_2 a_{ij})$ additional variables and $O(\log_2 a_{ij})$ constraints. The details of the representation are provided in [1]. The nonlinear inequalities (4.15) arising from the perspective reformulation of the Unit Commitment Problem are also representable with CQI as

$$a_{it}x_{it}^2 \leq z_{it}(b_{it}x_{it} - y_{it}).$$

The inequalities (4.20) from the SSSD problem from Section 4.5 are representable as rotated SOC constraints using the relation

$$z_{jk}y_{jk} + z_{jk}v_j - y_{jk}v_j \leq 0 \Leftrightarrow v_j^2 \leq (-z_{jk} + v_j)(y_{jk} + v_j).$$

For the mean-variance problem of Section 4.6, the constraints $y_i \geq x_i^2 \forall i \in N$ and the perspective version $y_i \geq x_i^2/z_i \forall i \in N$ can be placed in SOC format in the same fashion as the SQUFL problem in Section 4.1.

Using SOC software is generally preferable to using general NLP software for MINLP instances whose only nonlinearities can be put in SOC form. We will provide computational evidence for the improved performance of SOCP solvers over general NLP software in Section 6.

5.3. LP solvers. We next discuss how to use outer approximation cuts [9] to solve the perspective formulation via linear programming solvers. As current LP solvers are significantly faster than both NLP and SOCP solvers, this approach may offer significant advantages. We will discuss this idea using a simple MINLP where the nonlinearity is restricted to the objective function. Consider

$$\min_{(x,z) \in \mathbb{R}^n \times \mathbb{B}} \left\{ f(x) + cz \mid Ax \leq bz \right\},$$

where (i) $X = \{x \mid Ax \leq b\}$ is bounded (also implying $\{x \mid Ax \leq 0\} = \{0\}$), (ii) $f(x)$ is a convex function that is finite on X , and (iii) $f(0) = 0$. Under these assumptions, for any $\bar{x} \in X$ and subgradient $s \in \partial f(\bar{x})$, the following inequality

$$v \geq f(\bar{x}) + c + s^T(x - \bar{x}) + (c + f(\bar{x}) - s^T\bar{x})(z - 1) \quad (5.8)$$

is valid for the equivalent mixed integer program

$$\min_{(x,z,v) \in \mathbb{R}^n \times \mathbb{B} \times \mathbb{R}} \left\{ v \mid v \geq f(x) + cz, Ax \leq bz \right\}.$$

Inequality (5.8) is called the *perspective cut* and has been introduced by Frangioni and Gentile [11]. In their paper, Frangioni and Gentile use these cuts dynamically to build a tight formulation. It is possible to show [18] that perspective cuts are indeed outer approximation cuts for the perspective reformulation for this MINLP and therefore adding all (infinitely many) perspective cuts has the same strength as the perspective reformulation.

Furthermore, an interesting observation is that the perspective cuts can also be obtained by first building a linear outer approximation of the original nonlinear inequality $v \geq f(x) + cz$, and then strengthening it using a logical deductive argument. For example, in the SQUFL problem described in Section 4.1, the outer approximation of the inequality $y_{ij} \geq x_{ij}^2$ at a given point $(\bar{x}, \bar{y}, \bar{z})$ is

$$y_{ij} \geq 2(\bar{x}_{ij})x_{ij} - (\bar{x}_{ij})^2. \quad (5.9)$$

Using the observation that if $z_i = 0$, then $x_{ij} = y_{ij} = 0$, this inequality can be strengthened to

$$y_{ij} \geq 2(\bar{x}_{ij})x_{ij} - (\bar{x}_{ij})^2 z_i. \quad (5.10)$$

The inequality (5.10) is precisely the perspective cut (5.8) for this instance.

Following their work on perspective cuts, Frangioni and Gentile computationally compare using a LP solver (where perspective cuts are added dynamically) and using a second-order cone solver [13]. Based on their experiments on instances of the unit commitment problem and the portfolio optimization problem discussed earlier, they conclude that the dynamic (linear) approximation approach is significantly better than an SOC approach. The LP approach offers significant advantages, such as fast resolves in branch-and-bound, and the extensive array of cutting planes, branching rules, and heuristics that are available in powerful commercial MILP software. However, a dynamic cutting plane approach requires the use of the callable library of the solver software to add the cuts. For practitioners, an advantage of nonlinear automatic reformulation techniques is that they may be directly implemented in a modeling language.

Here, we offer a simple heuristic to obtain some of the strength of the perspective reformulation, while retaining the advantages of MILP software to solve the subproblem and an algebraic modeling language to formulate the instance. The heuristic works by choosing a set of points in advance and writing the perspective cuts using these points. This essentially gives a linear relaxation of the perspective formulation that uses piecewise linear under-approximations of the nonlinear functions. Solving this underapproximating MILP provides a lower bound on the optimal solution value of the MINLP. To obtain an upper bound, the integer variables may be fixed at the values found by the solution to the MILP, and a continuous NLP solved. We will demonstrate the effectiveness of this approach in the next section.

6. Computational results. The improvement in computational performance that can be obtained by using the perspective reformulation is exhibited on two families of instances, SQUFL (described in Section 4.1) and SSSD (described in Section 4.5). We also demonstrate the behavior of the various methodologies available (NLP, SOCP, LP) for solving the relaxations. When demonstrating the behavior of LP to solve the relaxations, we use the heuristic approach described at the end of Section 5.3. The reader interested in comparisons to a dynamic outer-approximation approach is referred to the work of Frangioni and Gentile [13].

6.1. Separable quadratic uncapacitated facility location. Random instances of SQUFL were generated similar to the instances of Günlük, Lee, and Weismantel [16]. For each facility $i \in M$, a location p_i is generated uniformly in $[0, 1]^2$ and the variable cost parameter was calculated

as $q_{ij} = 50\|p_i - p_j\|_2$. The fixed cost c_i of opening a facility is generated uniformly in $[1, 100]$. Ten random instances were created for values of $m \in \{20, 30\}$ and $n \in \{100, 150\}$. Thus, the instances solved had between 2100 and 4650 constraints, and between 2020 and 4530 variables, of which 20 or 30 (m) were binary variables. All instances were solved on an Intel Core 2 2.4GHz CPU, with a CPU time limit of 8 hours. Each instance was solved with four different methods.

1. The original formulation, was solved with CPLEX (v11.0) software for Mixed-Integer Quadratic Programming (MIQP);
2. The perspective reformulation, with the perspective inequalities put in rotated SOC form, was solved with the Mosek (v5.0) software for Mixed-Integer Second-Order Cone Programming (MIS-OCP);
3. The linear under-approximation using unstrengthened inequalities (5.9) was solved with the CPLEX (v11.0) software for Mixed-Integer Linear Programming (MILP); and
4. The linear under-approximation using the perspective cuts (5.10) was solved with the CPLEX (v11.0) software for MILP.

Optimization problems like SQUFL, that have a convex quadratic objective function and linear constraints, can be solved with a simplex-type pivoting method [20]. The pivoting method has significant advantages when used for branch-and-bound, as the solution procedure for child subproblems may be very effectively warmstarted using solution information from the parent node. CPLEX (v11.0) can use this pivoting-based method in its MIQP solver, which is why CPLEX was chosen as the solver for original formulation in method (1). The (rotated) SOC constraints (5.5) in the perspective reformulation can be directly specified to the solver Mosek via the GAMS modeling language. For this reason, we chose to use Mosek for solving the perspective reformulation in method (2). In methods (3) and (4), $|B| = 10$ breakpoints equally distributed between 0 and 1 were used to underestimate each nonlinear function. After solving the approximating MILP, the value of the integer variables was fixed and the resulting continuous relaxation (a convex quadratic program) was solved with CPLEX. Note that methods (3) and (4) are both implementations of the *heuristic* method described at the end of Section 5.3. Methods (1) and (2) are exact methods for solving the same problems. In order to make method (3) or (4) exact, the linearizations would have to be added dynamically as cutting planes.

Nonlinear and SOC Solvers: Table 1 shows the average performance of the two nonlinear formulations, methods (1) and (2), run on an Intel Core 2 2.4GHz CPU, with a CPU time limit of 8 hours. In the heading of the table, \bar{z}^* is the average optimal solution value, “# sol” denotes the number of instances out of 10 that were solved to optimality within the time limit, \bar{T} is the average CPU time required by the solver, and \bar{N} is the average number of nodes in the branch and bound search tree.

TABLE 1
Computational behavior of nonlinear formulations on SQUFL.

m	n	\bar{z}^*	Original			Perspective		
			#sol	\bar{T}	\bar{N}	#sol	\bar{T}	\bar{N}
20	100	408.31	10	307	6,165	10	18	37
20	150	508.38	10	807	7,409	10	33	29
30	100	375.86	10	4,704	67,808	10	33	53
30	150	462.69	7	16,607	96,591	10	56	40

From this experiment, the strong positive influence of the perspective reformulation is apparent—instances that cannot be solved in 8 hours with the original formulation are solved in around a minute using the strong perspective reformulation. In [18], the commercial solvers DICOPT [19] and BARON [28] were unable to solve small instances without the reformulation technique. This demonstrates that commercial MINLP solvers have yet to implement the perspective reformulation technique.

LP Solvers: Table 2 shows the results of solving the SQUFL instances using the two different linear approximations to the problem. In the table, the average of all lower bounds (obtained by solving the MILP outer-approximation to optimality) is given in the column \bar{z}_{lb} of the table. The average upper bound on z^* is given in the column \bar{z}_{ub} . The average CPU time (\bar{T}) and average number of nodes (\bar{N}) is also given in the table. The results demonstrate that strengthening the linear approximation of the nonlinear functions (the perspective cuts) significantly strengthens the formulation, as indicated by the significantly reduced number of nodes. Solving the root node linear relaxation with the piecewise linear approximation requires a significant time for both the strengthen and unstrengthened formulations, so the time improvements are not as great as in the nonlinear case.

TABLE 2
Computational behavior of linear formulations on SQUFL.

m	n	\bar{z}^*	Original			Perspective		
			\bar{T}	\bar{N}	\bar{T}	\bar{N}		
20	100	408.31	410.88	373.79	247	491	28	4
20	150	508.38	510.58	449.42	658	510	183	3
30	100	375.86	378.45	335.58	346	510	171	3
30	150	462.69	466.76	389.30	948	475	582	4

The lower bound provided by the solution to the approximating MILP is on average about 10% below the optimal solution value, and the upper bound found by fixing the integer variables is typically quite close to the

true optimal solution. In order to reduce the gap between lower and upper bounds in this heuristic approach, a finer approximation of the nonlinear function can be created by increasing $|B|$. Table 3 shows the results of an experiment where the instances were approximated with more linear inequalities, and the strengthened (perspective) reformulation was solved. In the table, $\bar{G}(\%) = 100(\bar{z}_{ub} - \bar{z}_{lb})/\bar{z}_{ub}$ denotes the average gap between lower and upper bounds in the heuristic approach. For these instances, $|B| = 50$ breakpoints is typically sufficient to prove that the solution obtained is within 1% of optimality. Note, however, the increase in CPU time required to solve the linear relaxations.

TABLE 3

Impact of number of piecewise linear approximation on gap and solution time.

m	n	$ B $	$\bar{G}(\%)$	\bar{T}	\bar{N}
20	100	10	9.12%	28	4
20	100	25	1.23%	122	2
20	100	50	0.31%	367	3
20	150	10	11.98%	183	3
20	150	25	1.45%	841	6
20	150	50	0.41%	2338	6
30	100	10	11.32%	171	3
30	100	25	1.35%	1000	9
30	100	50	0.39%	1877	5
30	150	10	16.6%	582	4
30	150	25	2.09%	1433	6
30	150	50	0.48%	3419	6

6.2. Stochastic service design. Instances of the SSSD were randomly created following the suggestions of Elhedhli [10]. The demand rate of each customer λ_i was uniformly distributed between 0.5 and 1.5. The lowest service rate μ_{j1} was uniformly distributed between $5|M|/8|N|$ and $7|M|/8|N|$. For $|K| = 3$, the remaining service levels were set to $\mu_{j2} = 2\mu_{j1}$, and $\mu_{j3} = 3\mu_{j1}$. The fixed cost for the lowest service level was uniformly distributed between 250 and 500, and to induce economies of scale, if $u_j = f_{j1}/\mu_{j1}$, then fixed costs for the higher service levels were set to $f_{j2} = u_j^{2/3}\mu_{j2}$ and $f_{j3} = u_j^{1/2}\mu_{j3}$. We use $t = 100$ for the queueing weight delay parameter, as these instances appear to be the most difficult computationally in the work of Elhedhli. [10]. Instances with $|M| = 15, 20, 25$ and $|N| = 4, 8$ were created. The MINLP formulations contained between 48 and 90 rows and between 89 and 257 columns, where the majority of the variables (between 72 and 224) were binary. The number of breakpoints used in the linearization for each instance was $|B| = 10$.

TABLE 4
Bonmin performance on SSSD instances.

		Without Perspective			With Perspective		
$ M $	$ N $	z^*	T (sec.)	# N	z^*	T (sec.)	# N
15	4	5.76	1161	21357	Failed after 236 nodes		
15	8	(9.37,9.41)	14400	224500	Failed after 91 nodes		
20	4	3.71	282	6342	Failed after 786 nodes		
20	8	(4.391,4.393)	14400	188987	Failed after 144 nodes		
25	4	2.98	238	3914	Failed after 235 nodes		
25	8	Failed after 2468 nodes			Failed after 85 nodes		

Each instance was solved six times with the following combination of formulation and software:

1. The original MINLP was solved with Bonmin (v0.9);
2. The perspective strengthened MINLP was solved with Bonmin;
3. The original instance was formulated using CQI and solved with Mosek (v5.0);
4. The perspective strengthened instance was formulated using CQI and solved with Mosek;
5. The linear under-approximation, not strengthened with perspective cuts, was solved with the CPLEX (v11.1) MILP solver. After fixing the integer variables to the solution of this problem, the continuous NLP problem was solved with IPOPT (v3.4);
6. The linear under-approximation, strengthened with perspective cuts, was solved with the CPLEX MILP solver. After fixing the integer variables to the solution of this problem, the continuous NLP problem was solved with IPOPT (v3.4).

NLP Solvers: Table 4 shows the results solving each instance, with and without the perspective strengthening, using the MINLP solver Bonmin. Bonmin uses the interior-point-based solver IPOPT to solve nonlinear relaxations. The table lists the optimal solution value (z^*) (or bounds on the best optimal solution), the CPU time required (T) in seconds, and the number of nodes evaluated (# N). A time limit of 4 hours was imposed.

In all cases, the NLP solver IPOPT failed at a node of the branch and bound tree with the message “**Error: Ipopt exited with error Restoration failed.**” For this instance, the NLP relaxation of SSSD (especially the perspective-enhanced NLP relaxation) appears difficult to solve. The fundamental issue is reliability not time, as when successful, all NLP relaxations solved in less than one second. We performed a small experiment designed to test the impact of the formulation and NLP software. In this experiment, four different nonlinear formulations of the perspective constraints were used, and the root node NLP relaxation was solved by

TABLE 5
Number of successful SSSD relaxation solutions (out of 10).

Solver	Formulation			
	F1	F2	F3	F4
Ipopt	0	10	10	10
Conopt	0	0	10	10
SNOPT	0	3	0	7

three different NLP packages: Ipopt (v3.4), Conopt (v3.14S), and SNOPT (v7.2-4). The root relaxation was also solved by Mosek (using the conic formulation) to obtain the true optimal solution value. The four different formulations of the perspective strengthening of the nonlinear constraints (4.17) were the following:

$$z_{jk}y_{jk} - z_{jk}v_j - v_jy_{jk} \leq 0, \quad (\text{F1})$$

$$z_{jk} - \frac{v_j}{1 + v_j/y_{jk}} \leq 0, \quad (\text{F2})$$

$$v_j^2 - (v_j - z_{jk})(v_j + y_{jk}) \leq 0, \quad (\text{F3})$$

$$\sqrt{4v_j^2 + (y_{jk} + z_{jk})^2} - 2v_j + y_{jk} - z_{jk} \leq 0. \quad (\text{F4})$$

In all cases, an initial iterate of

$$y_{jk} = 1/|K|, x_{ij} = 1/|J|, v_j = \frac{\sum_i \lambda_i x_{ij}}{\sum_k \mu_{jk} y_{jk} - \sum_i \lambda_i x_{ij}}, z_{jk} = \frac{v_j y_{jk}}{(1 + v_j)}$$

was used. Ten random instances of size $|M| = 100$, $|N| = 40$ were solved, and [Table 5](#) shows the number of instances for which the root node was correctly solved to (global) optimality for each formulation and software package. The results of this experiment indicate that modeling conic (perspective) inequalities in their convex form (F4) has an appreciably positive impact on the solution quality. The perspective results for Bonmin in [Table 4](#) were obtained with the formulation (F4), so even using the “best” formulation for the NLP solver was not sufficient to ensure the correct solution to the instance.

SOCOP Solvers: [Table 6](#) shows the results of solving the SSSD instances with the Mosek software (methods (3) and (4)). The table lists the time (T) in seconds, and the number of nodes ($\#N$). A time limit of 4 hours was imposed, and if the time limit was reached, indicated by T^* in the table, bounds on the optimal solution value are listed. In some cases, using the perspective reformulation has a very positive impact, while in other cases, the difference is minor. It is interesting to note the difference in behavior between Bonmin and Mosek without perspective strengthening. For example, Bonmin solved the $|M| = 15$, $|N| = 4$ instance in 21357 nodes,

while Mosek does not solve this instance in more than 1.9 million nodes. For these SSSD instances, Bonmin is able to add strong valid inequalities to improve performance, while Mosek does not add these inequalities.

TABLE 6
Mosek performance on SSSD instances.

		Without Perspective			With Perspective		
$ M $	$ N $	z^*	T	$\#N$	z^*	T	$\#N$
15	4	(4.88,5.76)	T^*	1.95M	5.76	250	26454
15	8	(8.53,9.41)	T^*	2.54M	(9.38,9.41)	T^*	2.06M
20	4	(2.77,3.71)	T^*	3.44M	3.71	52	12806
20	8	(4.31,4.39)	T^*	2.13M	(4.391,4.393)	T^*	1.50M
25	4	2.98	46	10128	2.98	19	3045
25	8	(6.143,6.146)	T^*	1.21M	(6.143,6.146)	T^*	1.18M

LP Solvers: Table 7 shows the results on the SSSD obtained when solving the linearized instances with CPLEX (methods (5) and (6)). In the table z_{lb} is the lower bound obtained by solving the MILP to optimality, and the time (T) and number of nodes ($\#N$) are also given for the search. The value z_{ub} is obtained by fixing the (optimal) integer solution and solving the NLP with Ipopt. The time require to solve the NLP is negligible. The results for the linear case are interesting. Specifically, strengthening the root relaxation by adding perspective cuts does not appear to help the computations in this case. For these instances, CPLEX is able to significantly improve the gap at the root node by adding its own cutting planes. Table 8 demonstrates the improvement in root lower bound value between the initial solve and final processing for both the strengthened and unstrengthened linear approximation, as well as the root lower bounds obtained from the nonlinear formulations. Note also the case $|M| = 25$, $|N| = 4$, where the solution obtained after fixing integer variables is far from optimal, demonstrating that one should not always expect to obtain a good solution from the linearization-based heuristic method.

7. Conclusions. The perspective reformulation is a tool to create strong relaxations of convex mixed integer nonlinear programs that have 0-1 variables to indicate special on-off logical relationships. The perspective reformulation can be derived as a special case of theorems of convex analysis or via techniques more familiar to MILP researchers: extended formulations, projection, and convex hulls of “simple” sets.

Many applications of MINLP could take advantage of this reformulation technique. In the applications described in this survey, the reformulated inequalities can be cast as second-order cone constraints, a transformation that can improve an instance’s solvability.

TABLE 7
Linear/CPLEX performance on SSSD instances.

		Without Perspective				With Perspective			
$ M $	$ N $	z_{lb}	z_{ub}	T	$\#N$	z_{lb}	z_{ub}	T	$\#N$
15	4	5.75	5.76	1.1	7586	5.75	5.76	1.1	7755
15	8	9.11	9.71	242	1.32M	9.11	9.71	818	4.46M
20	4	3.40	7.07	0.7	908	3.4	7.07	0.3	1517
20	8	4.37	4.41	810	5.59M	4.37	4.41	933	5.93M
25	4	2.64	17.2	0.7	486	2.64	17.2	0.8	541
25	8	6.13	6.16	374	1.75M	6.13	6.16	599	2.93M

TABLE 8
Initial and final root lower bounds for SSSD instances.

$ M $	$ N $	Without Perspective				With Perspective			
		Nonlinear	Linear		Nonlinear	Linear		Initial	Final
			Initial	Final		Initial	Final		
15	4	1.76	1.74	3.66	4.42	4.02	4.21		
15	8	3.16	2.06	5.68	6.75	6.11	6.41		
20	4	1.34	1.33	1.87	2.56	1.89	2.13		
20	8	1.64	1.55	2.41	2.93	2.86	2.99		
25	4	1.05	1.05	1.45	2.13	1.45	1.54		
25	8	2.18	2.15	3.53	4.27	3.97	4.20		

We hope this survey has achieved its goals of introducing a wider audience to the perspective reformulation technique, motivating software developers to consider automatic recognition of the structures required for the perspective reformulation, and spurring the research community to investigate additional simple sets occurring in practical MINLPs in the hope of deriving strong relaxations.

Acknowledgments. The authors would like to thank Jon Lee and Sven Leyffer for organizing the very fruitful meeting on MINLP at the Institute for Mathematics and its Applications (IMA) in November, 2008. The comments of Kevin Furman and Nick Sawaya were also helpful in preparing Section 2.2. The comments of two anonymous referees helped clarify the presentation and contribution.

REFERENCES

- [1] S. AKTÜRK, A. ATAMTÜRK, AND S. GÜREL, *A strong conic quadratic reformulation for machine-job assignment with controllable processing times*, Operations Research Letters, **37** (2009), pp. 187–191.

- [2] E. BALAS, *Disjunctive programming and a hierarchy of relaxations for discrete optimization problems*, SIAM Journal on Algebraic and Discrete Methods, **6** (1985), pp. 466–486.
- [3] A. BEN-TAL AND A. NEMIROVSKI, *Lectures on Modern Convex Optimization*, SIAM, 2001. MPS/SIAM Series on Optimization.
- [4] D. BERTSEKAS AND R. GALLAGER, *Data Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1987.
- [5] P. BONAMI, G. CORNUÉJOLS, AND H. HIJAZI, *Mixed integer non-linear programs with on/off constraints: Convex analysis and applications*, 2009. Poster Presentation at MIP 2009 Conference.
- [6] R. BOORSTYN AND H. FRANK, *Large-scale network topological optimization*, IEEE Transactions on Communications, **25** (1977), pp. 29–47.
- [7] B. BORCHERS AND J.E. MITCHELL, *An improved branch and bound algorithm for mixed integer nonlinear programs*, Computers & Operations Research, **21** (1994), pp. 359–368.
- [8] S. CERIA AND J. SOARES, *Convex programming for disjunctive optimization*, Mathematical Programming, **86** (1999), pp. 595–614.
- [9] M.A. DURAN AND I. GROSSMANN, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Mathematical Programming, **36** (1986), pp. 307–339.
- [10] S. ELHEDHLI, *Service system design with immobile servers, stochastic demand, and congestion*, Manufacturing & Service Operations Management, **8** (2006), pp. 92–97.
- [11] A. FRANGIONI AND C. GENTILE, *Perspective cuts for a class of convex 0–1 mixed integer programs*, Mathematical Programming, **106** (2006), pp. 225–236.
- [12] ———, *SDP diagonalizations and perspective cuts for a class of nonseparable MIQP*, Operations Research Letters, **35** (2007), pp. 181–185.
- [13] ———, *A computational comparison of reformulations of the perspective relaxation: SOCP vs. cutting planes*, Operations Research Letters, **24** (2009), pp. 105–113.
- [14] K. FURMAN, I. GROSSMANN, AND N. SAWAYA, *An exact MINLP formulation for nonlinear disjunctive programs based on the convex hull*, 2009. Presentation at 20th International Symposium on Mathematical Programming.
- [15] I. GROSSMANN AND S. LEE, *Generalized convex disjunctive programming: Nonlinear convex hull relaxation*, Computational Optimization and Applications (2003), pp. 83–100.
- [16] O. GÜNLÜK, J. LEE, AND R. WEISMANTEL, *MINLP strengthening for separable convex quadratic transportation-cost UFL*, Tech. Rep. RC24213 (W0703-042), IBM Research Division, March 2007.
- [17] O. GÜNLÜK AND J. LINDEROTH, *Perspective relaxation of mixed integer nonlinear programs with indicator variables*, Tech. Rep. RC24694 (W0811-076), IBM Research Division, November 2008.
- [18] ———, *Perspective relaxation of mixed integer nonlinear programs with indicator variables*, Mathematical Programming, Series B, **104** (2010), pp. 183–206.
- [19] G.R. KOCIS AND I.E. GROSSMANN, *Computational experience with DICOPT solving MINLP problems in process systems engineering*, Computers and Chemical Engineering, **13** (1989), pp. 307–315.
- [20] C.E. LEMKE, *Bimatrix equilibrium points and mathematical programming*, Management Science, **11** (1965), pp. 681–689.
- [21] S. LEYFFER AND J. LINDEROTH, *A practical guide to mixed integer nonlinear programming*, 2005. Short Course offered at SIAM Optimization Conference.
- [22] H.M. MARKOWITZ, *Portfolio selection*, Journal of Finance, **7** (1952), pp. 77–91.
- [23] *The mosek optimization tools manual. version 5.0 (revision 84)*, 2008. (www.mosek.com/)
- [24] A.F. PEROLD, *Large-scale portfolio optimization*, Management Science, **30** (1984), pp. 1143–1160.

- [25] R. STUBBS AND S. MEHROTRA, *A branch-and-cut method for 0-1 mixed convex programming*, Mathematical Programming, **86** (1999), pp. 515–532.
- [26] R.A. STUBBS, *Branch-and-Cut Methods for Mixed 0-1 Convex Programming*, PhD thesis, Northwestern University, December 1996.
- [27] J.F. STURM, *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*, Optimization Methods and Software, 11–12 (1999), pp. 625–653.
- [28] M. TAWARMALANI AND N.V. SAHINIDIS, *Global optimization of mixed integer nonlinear programs: A theoretical and computational study*, Mathematical Programming, **99** (2004), pp. 563–591.

PART II:

DISJUNCTIVE PROGRAMMING

GENERALIZED DISJUNCTIVE PROGRAMMING: A FRAMEWORK FOR FORMULATION AND ALTERNATIVE ALGORITHMS FOR MINLP OPTIMIZATION

IGNACIO E. GROSSMANN* AND JUAN P. RUIZ

Abstract. Generalized disjunctive programming (GDP) is an extension of the disjunctive programming paradigm developed by Balas. The GDP formulation involves Boolean and continuous variables that are specified in algebraic constraints, disjunctions and logic propositions, which is an alternative representation to the traditional algebraic mixed-integer programming formulation. After providing a brief review of MINLP optimization, we present an overview of GDP for the case of convex functions emphasizing the quality of continuous relaxations of alternative reformulations that include the big-M and the hull relaxation. We then review disjunctive branch and bound as well as logic-based decomposition methods that circumvent some of the limitations in traditional MINLP optimization. We next consider the case of linear GDP problems to show how a hierarchy of relaxations can be developed by performing sequential intersection of disjunctions. Finally, for the case when the GDP problem involves nonconvex functions, we propose a scheme for tightening the lower bounds for obtaining the global optimum using a combined disjunctive and spatial branch and bound search. We illustrate the application of the theoretical concepts and algorithms on several engineering and OR problems.

Key words. Disjunctive programming, Mixed-integer nonlinear programming, global optimization.

AMS(MOS) subject classifications.

1. Introduction. Mixed-integer optimization provides a framework for mathematically modeling many optimization problems that involve discrete and continuous variables. Over the last few years there has been a pronounced increase in the development of these models, particularly in process systems engineering [15, 21, 27].

Mixed-integer linear programming (MILP) methods and codes such as CPLEX, XPRESS and GUROBI have made great advances and are currently applied to increasingly larger problems. Mixed-integer nonlinear programming (MINLP) has also made significant progress as a number of codes have been developed over the last decade (e.g. DICOPT, SBB, α -ECP, Bonmin, FilMINT, BARON, etc.). Despite these advances, three basic questions still remain in this area: a) How to develop the “best” model?, b) How to improve the relaxation in these models?, c) How to solve nonconvex GDP problems to global optimality?

Motivated by the above questions, one of the trends has been to represent discrete and continuous optimization problems by models consisting of algebraic constraints, logic disjunctions and logic relations [32, 18]. The

*Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213
(grossmann@cmu.edu). NSF OCI-0750826.

basic motivation in using these representations is: a) to facilitate the modeling of discrete and continuous optimization problems, b) to retain and exploit the inherent logic structure of problems to reduce the combinatorics and to improve the relaxations, and c) to improve the bounds of the global optimum in nonconvex problems.

In this paper we provide an overview of Generalized Disjunctive Programming [32], which can be regarded as a generalization of disjunctive programming [3]. In contrast to the traditional algebraic mixed-integer programming formulations, the GDP formulation involves Boolean and continuous variables that are specified in algebraic constraints, disjunctions and logic propositions. We first address the solution of GDP problems for the case of convex functions for which we consider the big-M and the hull relaxation MINLP reformulations. We then review disjunctive branch and bound as well as logic-based decomposition methods that circumvent some of the MINLP reformulations. We next consider the case of linear GDP problems to show how a hierarchy of relaxations can be developed by performing sequential intersection of disjunctions. Finally, for the case when the GDP problem involves nonconvex functions, we describe a scheme for tightening the lower bounds for obtaining the global optimum using a combined disjunctive and spatial branch and bound search. We illustrate the application of the theoretical concepts and algorithms on several engineering and OR problems.

2. Generalized disjunctive programming. The most basic form of an MINLP problem is as follows:

$$\begin{aligned} \min \quad & Z = f(x, y) \\ \text{s.t.} \quad & g_j(x, y) \leq 0 \quad j \in J \\ & x \in X \quad y \in Y \end{aligned} \tag{MINLP}$$

where $f : R^n \rightarrow R^1$, $g : R^n \rightarrow R^m$ are differentiable functions, J is the index set of constraints, and x and y are the continuous and discrete variables, respectively. In the general case the MINLP problem will also involve nonlinear equations, which we omit here for convenience in the presentation. The set X commonly corresponds to a convex compact set, e.g. $X = \{x | x \in R^n, Dx \leq d, x^{lo} \leq x \leq x^{up}\}$; the discrete set Y corresponds to a polyhedral set of integer points, $Y = \{y | y \in Z^m, Ay \leq a\}$, which in most applications is restricted to 0-1 values, $y \in \{0, 1\}^m$. In most applications of interest the objective and constraint functions f , g are linear in y (e.g. fixed cost charges and mixed-logic constraints): $f(x, y) = c^T y + r(x)$, $g(x, y) = By + h(x)$. The derivation of most methods for MINLP assumes that the functions f and g are convex [14].

An alternative approach for representing discrete and continuous optimization problems is by using models consisting of algebraic constraints, logic disjunctions and logic propositions [4, 32, 40, 18, 19, 22]. This approach not only facilitates the development of the models by making the

formulation process intuitive, but it also keeps in the model the underlying logic structure of the problem that can be exploited to find the solution more efficiently. A particular case of these models is generalized disjunctive programming (GDP) [32] the main focus of this paper, and which can be regarded as a generalization of disjunctive programming [3]. Process Design [15] and Planning and Scheduling [27] are some of the areas where GDP formulations have shown to be successful.

2.1. Formulation. The general structure of a GDP can be represented as follows [32]:

$$\begin{aligned} \min \quad & Z = f(x) + \sum_{k \in K} c_k \\ \text{s.t.} \quad & g(x) \leq 0 \\ & \bigvee_{i \in D_k} \left[\begin{array}{l} Y_{ik} \\ r_{ik}(x) \leq 0 \\ c_k = \gamma_{ik} \end{array} \right] \quad k \in K \quad (\text{GDP}) \\ & \Omega(Y) = \text{True} \\ & x^{lo} \leq x \leq x^{up} \\ & x \in R^n, c_k \in R^1, Y_{ik} \in \{\text{True}, \text{False}\}, i \in D_k, k \in K \end{aligned}$$

where $f : R^n \rightarrow R^1$ is a function of the continuous variables x in the objective function, $g : R^n \rightarrow R^l$ belongs to the set of global constraints, the disjunctions $k \in K$, are composed of a number of terms $i \in D_k$, that are connected by the OR operator. In each term there is a Boolean variable Y_{ik} , a set of inequalities $r_{ik}(x) \leq 0$, $r_{ik} : R^n \rightarrow R^m$, and a cost variable c_k . If Y_{ik} is True, then $r_{ik}(x) \leq 0$ and $c_k = \gamma_{ik}$ are enforced; otherwise they are ignored. Also, $\Omega(Y) = \text{True}$ are logic propositions for the Boolean variables expressed in the conjunctive normal form $\Omega(Y) = \bigwedge_{t=1,2,\dots,T} \left[\bigvee_{Y_{ik} \in R_t} (Y_{ik}) \bigvee_{Y_{ik} \in Q_t} (\neg Y_{ik}) \right]$ where for each clause t , $t = 1, 2 \dots T$, R_t is the subset of Boolean variables that are non-negated, and Q_t is the subset of Boolean variables that are negated. As indicated in [36], we assume that the logic constraints $\bigvee_{i \in D_k} Y_{ik}$ are contained in $\Omega(Y) = \text{True}$.

There are three major cases that arise in problem (GDP): a) linear functions f , g and r ; b) convex nonlinear functions f , g and r ; c) non-convex functions f , g and r . Each of these cases require different solution methods.

2.2. Illustrative example. The following example illustrates how the GDP framework can be used to model the optimization of a simple process network shown in [Figure 1](#) that produces a product B by consuming a raw material A. The variables F represent material flows. The problem is to determine the amount of product to produce (F_8) with a selling price P_1 , the amount of raw material to buy (F_1) with a cost P_2 and the set

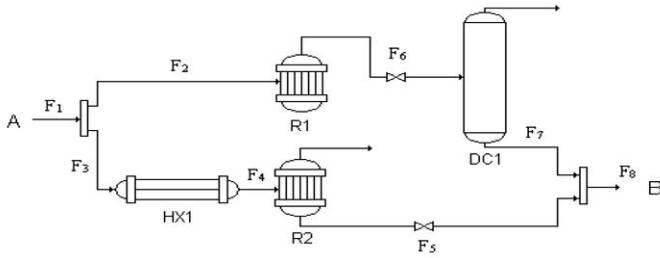


FIG. 1. Process network example.

of unit operations to use (i.e. $HX1$, $R1$, $R2$, $DC1$) with a cost $c_k = \gamma_k$, $k \in \{HX1, R1, R2, DC1\}$, in order to maximize the profit.

The generalized disjunctive program that represents the problem can be formulated as follows:

$$\max Z = P_1 F_8 - P_2 F_1 - \sum_{k \in K} c_k \quad (1)$$

$$s.t. \quad F_1 = F_3 + F_2 \quad (2)$$

$$F_8 = F_7 + F_5 \quad (3)$$

$$\left[\begin{array}{l} Y_{HX1} \\ F_4 = F_3 \\ c_{HX1} = \gamma_{HX1} \end{array} \right] \vee \left[\begin{array}{l} \neg Y_{HX1} \\ F_4 = F_3 = 0 \\ c_{HX1} = 0 \end{array} \right] \quad (4)$$

$$\left[\begin{array}{l} Y_{R2} \\ F_5 = \beta_1 F_4 \\ c_{R2} = \gamma_{R2} \end{array} \right] \vee \left[\begin{array}{l} \neg Y_{R2} \\ F_5 = F_4 = 0 \\ c_{R2} = 0 \end{array} \right] \quad (5)$$

$$\left[\begin{array}{l} Y_{R1} \\ F_6 = \beta_2 F_2 \\ c_{R1} = \gamma_{R1} \end{array} \right] \vee \left[\begin{array}{l} \neg Y_{R1} \\ F_6 = F_2 = 0 \\ c_{R1} = 0 \end{array} \right] \quad (6)$$

$$\left[\begin{array}{l} Y_{DC1} \\ F_7 = \beta_3 F_6 \\ c_{DC1} = \gamma_{DC1} \end{array} \right] \vee \left[\begin{array}{l} \neg Y_{DC1} \\ F_7 = F_6 = 0 \\ c_{DC1} = 0 \end{array} \right] \quad (7)$$

$$Y_{R2} \Leftrightarrow Y_{HX1} \quad (8)$$

$$Y_{R1} \Leftrightarrow Y_{DC1} \quad (9)$$

$$\begin{aligned} F_i \in R, c_k \in R^1, Y_k \in \{True, False\} & i \in \{1, 2, 3, 4, 5, 6, 7, 8\} \\ k \in \{HX1, R1, R2, DC1\} \end{aligned}$$

where (1) represents the objective function, (2) and (3) are the global constraints representing the mass balances around the splitter and mixer respectively, the disjunctions (4),(5),(6) and (7) represent the existence or non-existence of the unit operation k , $k \in \{HX1, R1, R2, DC1\}$ with their respective characteristic equations where β is the ratio between the inlet and outlet flows and (8) and (9) the logic propositions which enforce the selection of $DC1$ if and only if $R1$ is chosen and $HX1$ if and only if $R2$ is chosen. For the sake of simplicity we have presented here a simple linear model. In the actual application to a process problem there would be hundreds or thousands of nonlinear equations.

2.3. Solution methods.

2.3.1. MINLP reformulation. In order to take advantage of the existing MINLP solvers, GDPs are often reformulated as an MINLP by using either the big-M (BM) [28], or the Hull Relaxation (HR) [22] reformulation. The former yields:

$$\begin{aligned} \min \quad & Z = f(x) + \sum_{i \in D_k} \sum_{k \in K} \gamma_{ik} y_{ik} \\ \text{s.t.} \quad & g(x) \leq 0 \\ & r_{ik}(x) \leq M(1 - y_{ik}) \quad i \in D_k, \quad k \in K \quad (BM) \\ & \sum_{i \in D_k} y_{ik} = 1 \quad k \in K \\ & Ay \geq a \\ & x \in R^n, y_{ik} \in \{0, 1\}, \quad i \in D_k, \quad k \in K \end{aligned}$$

where the variable y_{ik} has a one to one correspondence with the Boolean variable Y_{ik} . Note that when $y_{ik} = 0$ and the parameter M is sufficiently large, the associated constraint becomes redundant; otherwise, it is enforced. Also, $Ay \geq a$ is the reformulation of the logic constraints in the discrete space, which can be easily accomplished as described in the work by Williams [46] and discussed in the work by Raman and Grossmann [32]. The hull reformulation yields,

$$\begin{aligned} \min \quad & Z = f(x) + \sum_{i \in D_k} \sum_{k \in K} \gamma_{ik} y_{ik} \\ \text{s.t.} \quad & x = \sum_{i \in D_K} \nu^{ik} \quad k \in K \\ & g(x) \leq 0 \\ & y_{ik} r_{ik} (\nu^{ik} / y_{ik}) \leq 0 \quad i \in D_k, \quad k \in K \quad (HR) \\ & 0 \leq \nu^{ik} \leq y_{ik} x^{up} \quad i \in D_k, \quad k \in K \\ & \sum_{i \in D_k} y_{ik} = 1 \quad k \in K \end{aligned}$$

$$Ay \geq a$$

$$x \in R^n, \nu^{ik} \in R^n, c_k \in R^1, y_{ik} \in \{0, 1\}, i \in D_k, k \in K.$$

As it can be seen, the HR reformulation is less intuitive than the BM. However, there is also a one to one correspondence between (GDP) and (HR). Note that the size of the problem is increased by introducing a new set of disaggregated variables ν^{ik} and new constraints. On the other hand, as proved in Grossmann and Lee [16] and discussed by Vecchietti, Lee and Grossmann [41], the HR formulation is at least as tight and generally tighter than the BM when the discrete domain is relaxed (i.e. $0 \leq y_{ik} \leq 1$, $k \in K, i \in D_k$). This is of great importance considering that the efficiency of the MINLP solvers heavily rely on the quality of these relaxations.

It is important to note that on the one hand the term $y_{ik} r_{ik} (\nu^{ik}/y_{ik})$ is convex if $r_{ik}(x)$ is a convex function. On the other hand the term requires the use of a suitable approximation to avoid singularities. Sawaya [36] proposed the following reformulation which yields an exact approximation at $y_{ik} = 0$ and $y_{ik} = 1$ for *any* value of ε in the interval $(0, 1)$, and the feasibility and convexity of the approximating problem are maintained:

$$y_{ik} r_{ik} (\nu^{ik}/y_{ik}) \approx ((1 - \varepsilon)y_{ik} + \varepsilon)r_{ik}(\nu^{ik}/((1 - \varepsilon)y_{ik} + \varepsilon)) - \varepsilon r_{ik}(0)(1 - y_{ik})$$

Note that this approximation assumes that $r_{ik}(x)$ is defined at $x = 0$ and that the inequality $0 \leq \nu^{ik} \leq y_{ik} x^{up}$ is enforced.

Methods that have addressed the solution of problem (MINLP) include the branch and bound method (BB) [17, 7, 38, 24], Generalized Benders Decomposition (GBD) [13], Outer-Approximation (OA) [10, 47, 11], LP/NLP based branch and bound [29, 8], and Extended Cutting Plane Method (ECP) [44, 45]. An extensive description of these methods is discussed in the review paper by Grossmann [14] while some implementation issues are discussed in Liberti et al. [25].

The number of computer codes for solving MINLP problems has increased in the last decade. The program DICOPT [43] is an MINLP solver that is available in the modeling system GAMS [9], and is based on the outer-approximation method with heuristics for handling nonconvexities. A similar code to DICOPT, AAOA, is available in AIMMS. Codes that implement the branch and bound method include the code MINLP-BB that is based on an SQP algorithm [24] and is available in AMPL, and the code SBB which is available in GAMS [9]. Both codes assume that the bounds are valid even though the original problem may be nonconvex. The code α -ECP that is available in GAMS implements the extended cutting plane method by Westerlund and Pettersson [44], including the extension by Westerlund and Pörn [45]. The open source code Bonmin [8] implements the branch and bound method, the outer-approximation and an extension of the LP/NLP based branch and bound method in one single framework. FilMINT [1] also implements a variant of the LP/NLP based branch and bound method. Codes for the global optimization that implement the

spatial branch and bound method include BARON [34], LINDOGlobal [26], and Couenne [5].

2.3.2. Logic-Based Methods. In order to fully exploit the logic structure of GDP problems, two other solution methods have been proposed for the case of convex nonlinear GDP, namely, the **Disjunctive Branch and Bound** method [22], which builds on the concept of Branch and Bound method by Beaumont [4] and the **Logic-Based Outer-Approximation** method [40].

The basic idea in the **disjunctive Branch and Bound** method is to directly branch on the constraints corresponding to particular terms in the disjunctions, while considering the hull relaxation of the remaining disjunctions. Although the tightness of the relaxation at each node is comparable with the one obtained when solving the HR reformulation with a MINLP solver, the size of the problems solved are smaller and the numerical robustness is improved.

For the case of **Logic-Based Outer-Approximation** methods, similar to the case of OA for MINLP, the main idea is to solve iteratively a master problem given by a linear GDP, which will give a lower bound of the solution and an NLP subproblem that will give an upper bound. As described in Turkay and Grossmann [40], for fixed values of the Boolean variables, $Y_{ik} = \text{True}$, $Y_{ik} = \text{False}$ with $i \neq k$, the corresponding NLP subproblem (SNLP) is as follows:

$$\begin{aligned} \min \quad & Z = f(x) + \sum_{k \in K} c_k \\ \text{s.t.} \quad & g(x) \leq 0 \\ & \left. \begin{array}{l} r_{ik}(x) \leq 0 \\ c_k = \gamma_{ik} \end{array} \right\} \text{for } Y_{ik} = \text{True} \quad i \in D_k, k \in K \quad (\text{SNLP}) \\ & x^{lo} \leq x \leq x^{up} \\ & x \in R^n, c_k \in R^1. \end{aligned}$$

It is important to note that only the constraints that belong to the active terms in the disjunction (i.e. associated Boolean variable $Y_{ik} = \text{True}$) are imposed. This leads to a substantial reduction in the size of the problem compared to the direct application of the traditional OA method on the MINLP reformulation. Assuming that L subproblems are solved in which sets of linearizations $\ell = 1, 2, \dots, L$ are generated for subsets of disjunction terms $L_{ik} = \{\ell | Y_{ik}^\ell = \text{True}\}$, one can define the following disjunctive OA master problem (MLGDP):

$$\begin{aligned} \min \quad & Z = \alpha + \sum_{k \in K} c_k \\ \text{s.t.} \quad & \left. \begin{array}{l} \alpha \geq f(x^\ell) + \nabla f(x^\ell)^T(x - x^\ell) \\ g(x^\ell) + \nabla g(x^\ell)^T(x - x^\ell) \leq 0 \end{array} \right\} \ell = 1, 2, \dots, L \end{aligned}$$

$$\bigvee_{i \in D_k} \left[\begin{array}{l} Y_{ik} \\ r_{ik}(x^\ell) + \nabla r_{ik}(x^\ell)(x - x^\ell) \leq 0 \quad \ell \in L_{ik} \\ c_k = \gamma_{ik} \end{array} \right] k \in K \quad (\text{MLGDP})$$

$$\Omega(Y) = \text{True}$$

$$x^{lo} \leq x \leq x^{up}$$

$$\alpha \in R^1, x \in R^n, c_k \in R^1, Y_{ik} \in \{\text{True}, \text{False}\}, i \in D_k, k \in K.$$

It should be noted that before applying the above master problem it is necessary to solve various subproblems (SNLP) for different values of the Boolean variables Y_{ik} so as to produce at least one linear approximation of each of the terms $i \in D_k$ in the disjunctions $k \in K$. As shown by Turkay and Grossmann [40] selecting the smallest number of subproblems amounts to solving a set covering problem, which is of small size and easy to solve. It is important to note that the number of subproblems solved in the initialization is often small since the combinatorial explosion that one might expect is in general limited by the propositional logic. This property frequently arises in Process Networks since they are often modeled by using two terms disjunctions where one of the terms is always linear (see remark below). Moreover, terms in the disjunctions that contain only linear functions need not be considered for generating the subproblems. Also, it should be noted that the master problem can be reformulated as an MILP by using the big-M or Hull reformulation, or else solved directly with a disjunctive branch and bound method.

REMARK. In the context of process networks the disjunctions in the (GDP) formulation typically arise for each unit i in the following form:

$$\left[\begin{array}{l} Y_i \\ r_i(x) \leq 0 \\ c_i = \gamma_i \end{array} \right] \vee \left[\begin{array}{l} \neg Y_i \\ B^i x = 0 \\ c_i = 0 \end{array} \right] \quad i \in I$$

in which the inequalities r_i apply and a fixed cost γ_i is incurred if the unit is selected (Y_i); otherwise ($\neg Y_i$) there is no fixed cost and a subset of the x variables is set to zero.

2.3.3. Example. We present here numerical results on an example problem dealing with the synthesis of a process network that was originally formulated by Duran and Grossmann [10] as an MINLP problem, and later by Turkay and Grossmann [40] as a GDP problem. [Figure 2](#) shows the superstructure that involves the possible selection of 8 processes. The Boolean variables Y_j denote the existence or non-existence of processes 1-8. The global optimal solution is $Z^*=68.01$, and consists of the selection of processes 2, 4, 6, and 8.

The model in the form of the GDP problem involves disjunctions for the selection of units, and propositional logic for the relationship of these units. Each disjunction contains the equation for each unit (these relax as convex inequalities). The model is as follows:

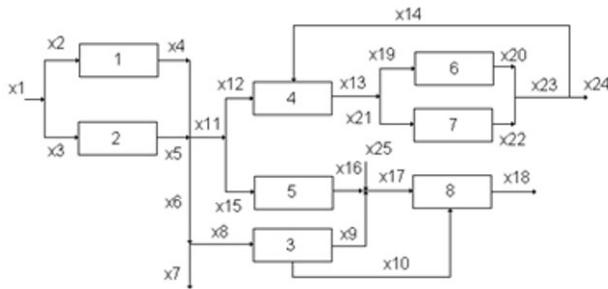


FIG. 2. Superstructure for process network.

Objective function:

$$\begin{aligned} \min Z = & c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7 + c_8 + x_2 - 10x_3 + x_4 \\ & - 15x_5 - 40x_9 + 15x_{10} + 15x_{14} + 80x_{17} - 65x_{18} + 25x_9 - 60x_{20} \\ & + 35x_{21} - 80x_{22} - 35x_{25} + 122 \end{aligned}$$

Material balances at mixing/splitting points:

$$\begin{aligned} x_3 + x_5 - x_6 - x_{11} &= 0 \\ x_{13} - x_{19} - x_{21} &= 0 \\ x_{17} - x_9 - x_{16} - x_{25} &= 0 \\ x_{11} - x_{12} - x_{15} &= 0 \\ x_6 - x_7 - x_8 &= 0 \\ x_{23} - x_{20} - x_{22} &= 0 \\ x_{23} - x_{14} - x_{24} &= 0 \end{aligned}$$

Specifications on the flows:

$$\begin{aligned} x_{10} - 0.8x_{17} &\leq 0 \\ x_{10} - 0.4x_{17} &\geq 0 \\ x_{12} - 5x_{14} &\leq 0 \\ x_{12} - 2x_{14} &\geq 0 \end{aligned}$$

Disjunctions:

$$\text{Unit 1: } \left[\begin{array}{l} Y_1 \\ e^{x_3} - 1 - x_2 \leq 0 \\ c_1 = 5 \end{array} \right] \vee \left[\begin{array}{l} \neg Y_1 \\ x_2 = x_3 = 0 \\ c_1 = 0 \end{array} \right]$$

$$\text{Unit 2: } \left[\begin{array}{l} Y_2 \\ e^{x_5/1.2} - 1 - x_4 \leq 0 \\ c_2 = 8 \end{array} \right] \vee \left[\begin{array}{l} \neg Y_2 \\ x_4 = x_5 = 0 \\ c_2 = 0 \end{array} \right]$$

$$\text{Unit 3: } \left[\begin{array}{l} Y_3 \\ 1.5x_9 - x_8 + x_{10} \leq 0 \\ c_3 = 6 \end{array} \right] \vee \left[\begin{array}{l} \neg Y_3 \\ x_8 = x_9 = x_{10} = 0 \\ c_3 = 0 \end{array} \right]$$

$$\begin{aligned}
 \text{Unit 4: } & \left[\begin{array}{l} Y_4 \\ 1.5(x_{12} + x_{14}) - x_{13} = 0 \\ c_4 = 10 \end{array} \right] \vee \left[\begin{array}{l} \neg Y_4 \\ x_{12} = x_{13} = x_{14} = 0 \\ c_4 = 0 \end{array} \right] \\
 \text{Unit 5: } & \left[\begin{array}{l} Y_5 \\ x_{15} - 2x_{16} = 0 \\ c_5 = 6 \end{array} \right] \vee \left[\begin{array}{l} \neg Y_5 \\ x_{15} = x_{16} = 0 \\ c_5 = 0 \end{array} \right] \\
 \text{Unit 6: } & \left[\begin{array}{l} Y_6 \\ e^{x_{20}/1.5} - 1 - x_{19} \leq 0 \\ c_6 = 7 \end{array} \right] \vee \left[\begin{array}{l} \neg Y_6 \\ x_{19} = x_{20} = 0 \\ c_6 = 0 \end{array} \right] \\
 \text{Unit 7: } & \left[\begin{array}{l} Y_7 \\ e^{x_{22}} - 1 - x_{21} \leq 0 \\ c_7 = 4 \end{array} \right] \vee \left[\begin{array}{l} \neg Y_7 \\ x_{21} = x_{22} = 0 \\ c_7 = 0 \end{array} \right] \\
 \text{Unit 8: } & \left[\begin{array}{l} Y_8 \\ e^{x_{18}} - 1 - x_{10} - x_{17} \leq 0 \\ c_8 = 5 \end{array} \right] \vee \left[\begin{array}{l} \neg Y_8 \\ x_{10} = x_{17} = x_{18} = 0 \\ c_8 = 0 \end{array} \right]
 \end{aligned}$$

Propositional Logic:

$$\begin{aligned}
 Y_1 &\Rightarrow Y_3 \vee Y_4 \vee Y_5; Y_2 \Rightarrow Y_3 \vee Y_4 \vee Y_5; Y_3 \Rightarrow Y_1 \vee Y_2; Y_3 \Rightarrow Y_8 \\
 Y_4 &\Rightarrow Y_1 \vee Y_2; Y_4 \Rightarrow Y_6 \vee Y_7; Y_5 \Rightarrow Y_1 \vee Y_2; Y_5 \Rightarrow Y_8 \\
 Y_6 &\Rightarrow Y_4; Y_7 \Rightarrow Y_4 \\
 Y_8 &\Rightarrow Y_3 \vee Y_5 \vee (\neg Y_3 \wedge \neg Y_5)
 \end{aligned}$$

Specifications:

$$Y_1 \leq Y_2; Y_4 \leq Y_5; Y_6 \leq Y_7$$

Variables:

$$x_j, c_i \geq 0, Y_i = \{\text{True}, \text{False}\} \quad i = 1, 2 \dots 8, j = 1, 2 \dots 25.$$

Table 1 shows a comparison between the three solution approaches presented before. Master and NLP represent the number of MILP master problems and NLP subproblems solved to find the solution. It should be noted that the Logic-Based Outer-Approximation method required solving only three NLP subproblems to initialize the master problem (MGDLP), which was reformulated as an MILP using the hull relaxation reformulation.

2.4. Special cases.

2.4.1. Linear generalized disjunctive programming. A particular class of GDP problems arises when the functions in the objective and constraints are linear. The general formulation of a linear GDP as described by Raman and Grossmann [32] is as follows:

TABLE 1
Results using different GDP solution methods.

	Outer-Approximation*	Disjunctive B&B	Logic-Based OA**
NLP	2	5	4
Master	2	0	1

* Solved with DICOPT through EMP (GAMS).

** Solved with LOGMIP (GAMS).

$$\begin{aligned}
 \min \quad & Z = d^T x + \sum_k c_k \\
 \text{s.t.} \quad & Bx \leq b \\
 & \bigvee_{i \in D_k} \left[\begin{array}{l} Y_{ik} \\ A_{ik}x \leq a_{ik} \\ c_k = \gamma_{ik} \end{array} \right] \quad k \in K \\
 & \Omega(Y) = \text{True} \\
 & x^{lo} \leq x \leq x^{up} \\
 & x \in R^n, c_k \in R^1, Y_{ik} \in \{\text{True}, \text{False}\}, i \in D_k, k \in K.
 \end{aligned} \tag{LGDP}$$

The big-M formulation reads:

$$\begin{aligned}
 \min \quad & Z = d^T x + \sum_{i \in D_k} \sum_{k \in K} \gamma_{ij} y_{ik} \\
 \text{s.t.} \quad & Bx \leq b \\
 & A_{ik}x \leq a_{ik} + M(1 - y_{ik}) \quad i \in D_k, k \in K \quad (\text{LBM}) \\
 & \sum_{i \in D_k} y_{ik} = 1 \quad k \in K \\
 & Ay \geq a \\
 & x \in R^n, y_{ik} \in \{0, 1\}, i \in D_k, k \in K
 \end{aligned}$$

while the HR formulation reads:

$$\begin{aligned}
 \min \quad & Z = d^T x + \sum_{i \in D_k} \sum_{k \in K} \gamma_{ij} y_{ik} \\
 \text{s.t.} \quad & x = \sum_{i \in D_K} \nu^{ik} \quad k \in K \\
 & Bx \leq b \\
 & A_{ik} \nu^{ik} \leq a_{ik} y_{ik} \quad i \in D_k, k \in K \\
 & 0 \leq \nu^{ik} \leq y_{ik} U_v \quad i \in D_k, k \in K \\
 & \sum_{i \in D_k} y_{ik} = 1 \quad k \in K \\
 & Ay \geq a \\
 & x \in R^n, \nu_{ik} \in R^n, c_k \in R^1, y_{ik} \in \{0, 1\}, i \in D_k, k \in K.
 \end{aligned} \tag{LHR}$$

As a particular case of a GDP, LGDPs can be solved using MIP solvers applied on the LBM or LHR reformulations. However, as described in the

work of Sawaya and Grossmann [35] two issues may arise. Firstly, the continuous relaxation of LBM is often weak, leading to a large number of nodes enumerated in the branch and bound procedure. Secondly, the increase in the size of LHR due to the disaggregated variables and new constraints may not compensate for the strengthening obtained in the relaxation, resulting in a high computational effort. In order to overcome these issues, Sawaya and Grossmann [35] proposed a cutting plane methodology that consists in the generation of cutting planes obtained from the LHR and used to strengthen the relaxation of LBM. It is important to note, however, that in the last few years, MIP solvers have improved significantly in the use of the problem structure to reduce automatically the size of the formulation. As a result the emphasis should be placed on the strength of the relaxations rather than on the size of formulations. With this in mind, we present next the last developments in linear GDPs.

Sawaya [36] proved that any Linear Generalized Disjunctive Program (LGDP) that involves Boolean and continuous variables can be equivalently formulated as a Disjunctive Program (DP), that only involves continuous variables. This means that we are able to exploit the wealth of theory behind DP from Balas [2, 3] in order to solve LGDP more efficiently.

One of the properties of disjunctive sets is that they can be expressed in many different equivalent forms. Among these forms, two extreme ones are the Conjunctive Normal Form (CNF), which is expressed as the intersection of elementary sets (i.e. sets that are the union of half spaces), and the Disjunctive Normal Form (DNF), which is expressed as the union of polyhedra. One important result in Disjunctive Programming Theory, as presented in the work of Balas [3], is that we can systematically generate a set of equivalent DP formulations going from the CNF to the DNF by using an operation called **basic step** (Theorem 2.1 [3]), which preserves regularity. A basic step is defined as follows. Let F be the disjunctive set in regular form (RF) given by $F = \bigcap_{j \in T} S_j$ where $S_j = \bigcup_{i \in Q_j} P_i$, P_i a polyhedron, $i \in Q_j$. For $k, l \in T$, $k \neq l$, a basic step consists in replacing $S_k \cap S_l$ with $S_{kl} = \bigcup_{\substack{i \in Q_k \\ j \in Q_l}} (P_i \cap P_j)$. Note that a basic step involves intersecting a given pair of disjunctions S_k and S_l .

Although the formulations obtained after the application of basic steps on the disjunctive sets are equivalent, their *continuous relaxations* are not. We denote the continuous relaxation of a disjunctive set $F = \bigcap_{j \in T} S_j$ in regular form where each S_j is a union of polyhedra, as the *hull-relaxation* of F (or $h - rel F$). Here $h - rel F := \bigcap_{j \in T} clconv S_j$ and $clconv S_j$ denotes the closure of the convex hull of S_j . That is, if $S_j = \bigcup_{i \in Q_j} P_i$, $P_i = \{x \in R^n, A^i x \leq b^i\}$, then $clconv S_j$ is given by, $x = \sum_{i \in Q_j} \nu^i$, $\lambda_i \geq 0$, $\sum_{i \in Q_j} \lambda_i = 1$, $A^i \nu^i \leq b^i \lambda_i$, $i \in Q_j$. Note that the convex hull of F is

in general different from its hull-relaxation.

As described by Balas (Theorem 4.3 [3]), the application of a basic step on a disjunctive set leads to a new disjunctive set whose relaxation is at least as tight, if not tighter, as the former. That is, for $i = 0, 1, \dots, t$ let $F_i = \bigcap_{j \in T_i} S_j$ be a sequence of regular forms of a disjunctive set, such that: i) F_0 is in CNF, with $P_0 = \bigcap_{j \in T_0} S_j$, ii) F_t is in DNF, iii) for $i = 1, \dots, t$, F_i is obtained from F_{i-1} by a basic step. Then $h - \text{rel } F_0 \supseteq h - \text{rel } F_1 \supseteq \dots \supseteq h - \text{rel } F_t$. As shown by Sawaya [36], this leads to a procedure to find MIP reformulations that are often tighter than the traditional LHR.

To illustrate let us consider the following example:

$$\begin{aligned} \min \quad & Z = x_2 \\ \text{s.t.} \quad & 0.5x_1 + x_2 \leq 1 \\ & \left[\begin{array}{c} Y_1 \\ x_1 = 0 \\ x_2 = 0 \end{array} \right] \vee \left[\begin{array}{c} \neg Y_1 \\ x_1 = 1 \\ 0 \leq x_2 \leq 1 \end{array} \right] \quad (\text{LGDP1}) \\ & 0 \leq x_1, x_2 \leq 1 \\ & x_1, x_2 \in R, Y_1 \in \{\text{True}, \text{False}\}. \end{aligned}$$

An equivalent formulation can be obtained by the application of a basic step between the global constraint (or one term disjunction) $0.5x_1 + x_2 \leq 1$ and the two-term disjunction.

$$\begin{aligned} \min \quad & Z = x_2 \\ \text{s.t.} \quad & \left[\begin{array}{c} Y_1 \\ x_1 = 0 \\ x_2 = 0 \\ 0.5x_1 + x_2 \leq 1 \end{array} \right] \vee \left[\begin{array}{c} \neg Y_1 \\ x_1 = 1 \\ 0 \leq x_2 \leq 1 \\ 0.5x_1 + x_2 \leq 1 \end{array} \right] \quad (\text{LGDP2}) \\ & 0 \leq x_1, x_2 \leq 1 \\ & x_1, x_2 \in R, Y_1 \in \{\text{True}, \text{False}\}. \end{aligned}$$

As it can be seen in Figure 3, the hull relaxation of the later formulation is tighter than the original leading to a stronger lower bound.

Example. Strip Packing Problem (Hifi, 1998). We apply the new approach to obtain stronger relaxations on a set of instances for the Strip Packing Problem. Given a set of small rectangles with width H_i and length L_i and a large rectangular strip of fixed width W and unknown length L . The problem is to fit the small rectangles on the strip (without rotation and overlap) in order to minimize the length L of the strip.

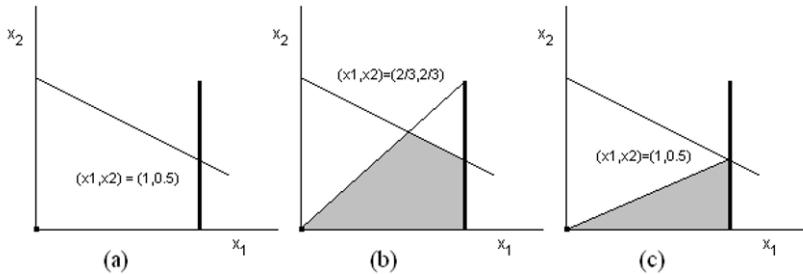


FIG. 3. *a*-Projected feasible region of LGDP1 , *b*-Projected feasible region of relaxed LGDP1 , *c*-Projected feasible region of relaxed LGDP2.

The LGDP for this problem is presented below: [36]

$$\begin{aligned}
 \min \quad & Z = lt \\
 \text{s.t.} \quad & lt \geq x_i + L_i \quad \forall i \in N \\
 & \left[\begin{array}{l} Y_{ij}^1 \\ x_i + L_i \leq x_j \end{array} \right] \vee \left[\begin{array}{l} Y_{ij}^2 \\ x_j + L_j \leq x_i \end{array} \right] \vee \\
 & \left[\begin{array}{l} Y_{ij}^3 \\ y_i - H_i \geq y_j \end{array} \right] \vee \left[\begin{array}{l} Y_{ij}^4 \\ y_j - H_j \geq y_i \end{array} \right] \forall i, j \in N, i < j \\
 & x_i \leq UB_i - L_i \quad \forall i \in N \\
 & H_i \leq y_i \leq W \quad \forall i \in N \\
 & lt, x_i, y_i \in R_+^1, Y_{ij}^{1,2,3,4} \in \{\text{True}, \text{False}\} \quad \forall i, j \in N, i < j.
 \end{aligned}$$

In Table 2, the approach using basic steps to obtain stronger relaxations is compared with the original formulation.

TABLE 2
Comparison of sizes and lower bounds between original and new MIP reformulations.

Instance	Formulation without Basic Steps				Formulation with Basic Steps			
	Vars	0-1	Constr.	LB	Vars	0-1	Constr.	LB
4 Rectang.	102	24	143	4	170	24	347	8
25 Rectang.	4940	1112	7526	9	5783	1112	8232	27
31 Rectang.	9716	2256	14911	10.64	11452	2256	15624	33

It is important to note that although the size of the reformulated MIP is significantly increased when applying basic steps, the LB is greatly improved.

2.4.2. Nonconvex generalized disjunctive programs. In general, some of the functions f , r_{ik} or g might be nonconvex, giving rise to a nonconvex GDP problem. The direct application of traditional algorithms to solve the reformulated MINLP in this case, such as Generalized Benders Decomposition (GBD) [6, 13] or Outer-Approximation (OA) [43] may fail to find the global optimum since the solution of the NLP subproblem may correspond to a local optimum and the cuts in the master problem may not be valid. Therefore, specialized algorithms should be used in order to find the global optimum [20, 39]. With this aim in mind, Lee and Grossmann [23] proposed the following two-level branch and bound algorithm.

The first step in this approach is to introduce convex underestimators of the nonconvex functions in the original nonconvex GDP. This leads to:

$$\begin{aligned} \min \quad & Z = \bar{f}(x) + \sum_{i \in D_k} \sum_{k \in K} \gamma_{ij} y_{ik} \\ \text{s.t.} \quad & \bar{g}(x) \leq 0 \\ & \bigvee_{i \in D_k} \left[\begin{array}{l} Y_{ik} \\ \bar{r}_{ik}(x) \leq 0 \\ c_k = \gamma_{ik} \end{array} \right] \quad k \in K \quad (\text{RGDPNC}) \\ & \Omega(Y) = \text{True} \\ & x^{lo} \leq x \leq x^{up} \\ & x \in R^n, c_k \in R^1, Y_{ik} \in \{\text{True}, \text{False}\}, i \in D_k, k \in K \end{aligned}$$

where \bar{f} , \bar{r}_{ik} , \bar{g} are convex and the following inequalities are satisfied $\bar{f}(x) \leq f(x)$, $\bar{r}_{ik}(x) \leq r_{ik}(x)$, $\bar{g}(x) \leq g(x)$. Note that suitable convex underestimators for these functions can be found in Tawarmalani and Sahinidis [39].

The feasible region of (RGDPNC) can be relaxed by replacing each disjunction by its convex hull. This relaxation yields the following convex NLP:

$$\begin{aligned} \min \quad & Z = \bar{f}(x) + \sum_{i \in D_k} \sum_{k \in K} \gamma_{ij} y_{ik} \\ \text{s.t.} \quad & x = \sum_{i \in D_K} \nu^{ik} \quad k \in K \\ & \bar{g}(x) \leq 0 \\ & y_{ik} \bar{r}_{ik}(\nu^{ik}/y_{ik}) \leq 0 \quad i \in D_k, k \in K \quad (\text{RGDPRNC}) \\ & 0 \leq \nu^{ik} \leq y_{ik} x^{up} \quad i \in D_k, k \in K \\ & \sum_{i \in D_k} y_{ik} = 1 \quad k \in K \\ & Ay \geq a \\ & x \in R^n, \nu_{ik} \in R^n, c_k \in R^1, y_{ik} \in [0, 1], i \in D_k, k \in K. \end{aligned}$$

As proved in Lee and Grossmann [23] the solution of this NLP formulation leads to a lower bound of the global optimum.

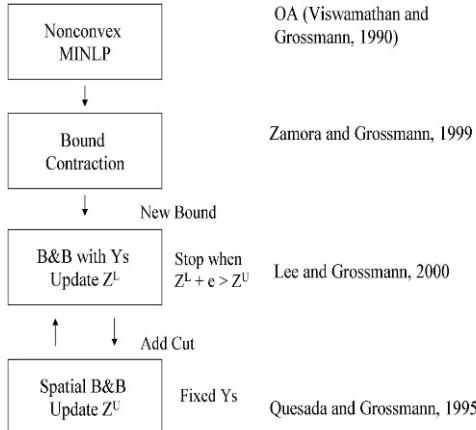


FIG. 4. Steps in global optimization algorithm.

The second step consists in using the above relaxation to predict lower bounds within a spatial branch and bound framework. The main steps in this implementation are described in Figure 4. The algorithm starts by obtaining a local solution of the nonconvex GDP problem by solving the MINLP reformulation with a local optimizer (e.g. DICOPT), which provides an upper bound of the solution (Z^U). Then, a bound contraction procedure is performed as described by Zamora and Grossmann [48]. Finally, a partial branch and bound method is used on *RGDPNC* as described in Lee and Grossmann [23] that consists in only branching on the Boolean variables until a node with all the Boolean variables fixed is reached. At this point a spatial branch and bound procedure is performed as described in Quesada and Grossmann [30].

While the method proved to be effective in solving several problems, a major question is whether one might be able to obtain stronger lower bounds to improve the computational efficiency.

Recently, Ruiz and Grossmann [33] proposed an enhanced methodology that builds on the work of Sawaya [36] to obtain stronger relaxations. The basic idea consists in relaxing the nonconvex terms in the GDP using valid linear over- and underestimators previous to the application of basic steps. This leads to a new linear GDP whose continuous relaxation is tighter and valid for the original nonconvex GDP problem. The implementation of basic steps is not trivial. Therefore, Ruiz and Grossmann [33] proposed a set of rules that aims at keeping the formulation small while improving the relaxation. Among others, it was shown that intersecting the global constraints with the disjunctions leads to a linear GDP with the same number of disjuncts but a stronger relaxation.

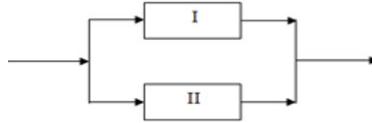


FIG. 5. Two reactor network.

The following example illustrates the idea behind this approach to obtain a stronger relaxation in a simple nonconvex GDP. Figure 5 shows a small superstructure consisting of two reactors, each characterized by a flow-conversion curve, a conversion range for which it can be designed, and its corresponding cost as can be seen in Table 3. The problem consists in choosing the reactor and conversion that maximize the profit from sales of the product considering that there is a limit on the demand. The characteristic curve of the reactors is defined as $F = aX + b$ in the range of conversions $[X^{lo}, X^{up}]$ where F and X are the flow of raw material and conversion respectively. The price of the product is given by $\theta = 2$, the cost of the raw material is given by $\lambda = 0.2$ and the limit in the demand by $d = 2$.

TABLE 3
Reactor characteristics.

Reactor	Curve		Range		Cost
	a	b	X^{lo}	X^{up}	
I	-8	9	0.2	0.95	2.5
II	-10	15	0.7	0.99	1.5

The bilinear GDP model, which maximizes the profit, can be stated as follows:

$$\begin{aligned}
 \max \quad & Z = \theta FX - \gamma F - CP \\
 \text{s.t.} \quad & FX \leq d \\
 & \left[\begin{array}{l} Y_{11} \\ F = \alpha_1 X + \beta_1 \\ X_1^{lo} \leq X \leq X_1^{up} \\ CP = Cp_1 \end{array} \right] \vee \left[\begin{array}{l} Y_{21} \\ F = \alpha_2 X + \beta_2 \\ X_2^{lo} \leq X \leq X_2^{up} \\ CP = Cp_2 \end{array} \right] \quad (GDP1_{NC}) \\
 & Y_{11} \veebar Y_{21} = \text{True} \\
 & X, F, CP \in R^1, F^{lo} \leq F \leq F^{up}, Y_{11}, Y_{21} \in \{\text{True}, \text{False}\}
 \end{aligned}$$

The associated linear GDP relaxation is obtained by replacing the bilinear term, FX , using the McCormick convex envelopes:

$$\max \quad Z = \theta P - \gamma F - CP$$

$$s.t. \quad P \leq d$$

$$P \leq FX^{lo} + F^{up}X - F^{up}X^{lo} \quad (GDP1_{RLP0})$$

$$P \leq FX^{up} + F^{lo}X - F^{lo}X^{up}$$

$$P \geq FX^{lo} + F^{lo}X - F^{lo}X^{lo}$$

$$P \geq FX^{up} + F^{up}X - F^{up}X^{up}$$

$$\left[\begin{array}{l} Y_{11} \\ F = \alpha_1 X + \beta_1 \\ X_1^{lo} \leq X \leq X_1^{up} \\ CP = Cp_1 \end{array} \right] \vee \left[\begin{array}{l} Y_{21} \\ F = \alpha_2 X + \beta_2 \\ X_2^{lo} \leq X \leq X_2^{up} \\ CP = Cp_2 \end{array} \right]$$

$$Y_{11} \veebar Y_{21} = True$$

$$X, F, CP \in R^1, F^{lo} \leq F \leq F^{up}, Y_{11}, Y_{21} \in \{True, False\}.$$

Intersecting the improper disjunctions given by the inequalities of the relaxed bilinear term with the only proper disjunction (i.e. by applying five basic steps), we obtain the following GDP formulation,

$$\max \quad Z = \theta P - \gamma F - CP \quad (GDP1_{RLP1})$$

$$s.t.$$

$$\left[\begin{array}{l} Y_{11} \\ P \leq d \\ P \leq FX^{up} + F^{lo}X - F^{lo}X^{up} \\ P \leq FX^{lo} + F^{up}X - F^{up}X^{lo} \\ P \geq FX^{lo} + F^{lo}X - F^{lo}X^{lo} \\ P \geq FX^{up} + F^{up}X - F^{up}X^{up} \\ F = \alpha_1 X + \beta_1 \\ X_1^{lo} \leq X \leq X_1^{up} \\ CP = Cp_1 \end{array} \right] \vee \left[\begin{array}{l} Y_{21} \\ P \leq d \\ P \leq FX^{up} + F^{lo}X - F^{lo}X^{up} \\ P \leq FX^{lo} + F^{up}X - F^{up}X^{lo} \\ P \geq FX^{lo} + F^{lo}X - F^{lo}X^{lo} \\ P \geq FX^{up} + F^{up}X - F^{up}X^{up} \\ F = \alpha_2 X + \beta_2 \\ X_2^{lo} \leq X \leq X_2^{up} \\ CP = Cp_2 \end{array} \right]$$

$$Y_{11} \veebar Y_{21} = True$$

$$X, F, CP \in R^1, F^{lo} \leq F \leq F^{up}, Y_{11}, Y_{21} \in \{True, False\}.$$

[Figure 6](#) shows the actual feasible region of $(GDP1_{NC})$ and the projection on the $F - X$ space of the hull relaxations of $(GDP1_{RLP0})$ and $(GDP1_{RLP1})$, where clearly the feasible space in $(GDP1_{RLP1})$ is tighter than in $(GDP1_{RLP0})$. Notice that in this case the choice of reactor II is infeasible.

Example. Water treatment network . This example corresponds to a synthesis problem of a distributed wastewater multicomponent network (See [Figure 7](#)), which is taken from Galan and Grossmann [12]. Given a set of process liquid streams with known composition, a set of technologies for the removal of pollutants, and a set of mixers and splitters, the objective is to find the interconnections of the technologies and their flowrates to meet the specified discharge composition of pollutant at minimum total

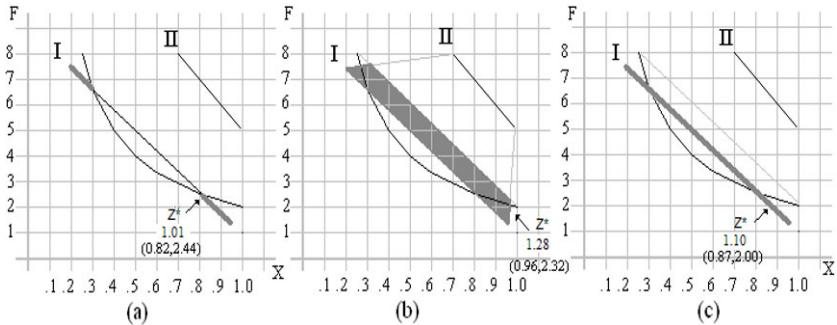


FIG. 6. a-Projected feasible region of $GDP1_{NC}$, b-Projected feasible region of relaxed $GDP1_{RLP0}$, c-Projected feasible region of relaxed $GDP1_{RLP1}$.

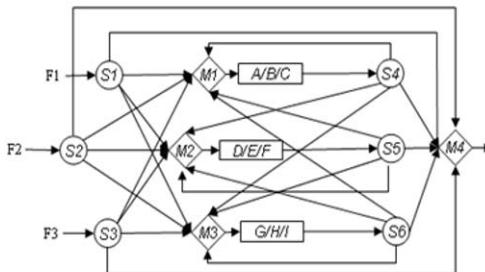


FIG. 7. Water treatment superstructure.

cost. Discrete choices involve deciding what equipment to use for each treatment unit.

Lee and Grossmann [23] formulated this problem as the following nonconvex GDP problem:

$$\begin{aligned}
 \min \quad & Z = \sum_{k \in PU} CP_k \\
 \text{s.t.} \quad & f_k^j = \sum_{i \in M_k} f_i^j \quad \forall jk \in MU \\
 & \sum_{i \in S_k} f_i^j = f_k^j \quad \forall jk \in SU \\
 & \sum_{i \in S_k} \zeta_i^k = 1 \quad k \in SU \\
 & f_i^j = \zeta_i^k f_k^j \quad \forall j \quad i \in S_k \quad k \in SU \\
 & \bigg. \quad \begin{array}{l} YP_k^h \\ f_i^j = \beta_k^{jh} f_{i'}^j, i \in OPU_k, i' \in IPU_k, \forall j \\ F_k = \sum_j f_i^j, i \in OPU_k \\ CP_k = \partial_{ik} F_k \end{array} \quad \bigg] \quad k \in PU
 \end{aligned}$$

$$\begin{aligned}
0 \leq \zeta_i^k \leq 1 & \quad \forall i, k \\
0 \leq f_i^j, f_k^j & \quad \forall i, j, k \\
0 \leq CP_k & \quad \forall k \\
YP_k^h \in \{True, False\} & \quad \forall h \in D_k \quad \forall k \in PU.
\end{aligned}$$

The problem involves 9 discrete variables and 114 continuous variables with 36 bilinear terms.

As it can be seen in [Table 4](#), an improved lower bound was obtained (i.e. 431.9 vs 400.66) which is a direct indication of the reduction of the relaxed feasible region. The column “Best Lower Bound”, can be used as an indicator of the performance of the proposed set of rules to apply basic steps. Note that the lower bound obtained in this new approach is the same as the one obtained by solving the relaxed DNF, which is quite remarkable. A further indication of tightening is shown in [Table 5](#) where numerical results of the branch and bound algorithm proposed in section 6 are presented. As it can be seen the number of nodes that the spatial branch and bound algorithm requires before finding the global solution is significantly reduced.

TABLE 4
Comparison of lower bounds obtained using different relaxations.

Global Optimum	Lower Bound (Lee and Grossmann)	Lower Bound (Ruiz and Grossmann)	Best Lower Bound
1214.87	400.66	431.9	431.9

TABLE 5
Performance using different relaxations within a spatial B&B.

Lee and Grossmann			Ruiz and Grossmann		
Nodes	Bounding %	Time(sec)	Nodes	Bounding %	Time(sec)
408	8	176	130	16	115

[Table 6](#) shows the size of the LP relaxation obtained in each case. Note that although the proposed methodology leads to a significant increase in the size of the formulation, this is not translated proportionally to the solution time of the resulting LP. This behavior can be understood by considering that in general, the LP pre-solver will take advantage of the particular structures of these LPs.

3. Conclusions. In this paper we have provided an overview of the Generalized Disjunctive Programming Framework. We presented different solution strategies that exploit the underlying logic structure of the formulations with particular focus on how to develop formulations that lead to

TABLE 6
Size of the LP relaxation for example problems.

Lee and Grossmann		Ruiz and Grossmann	
Constraints	Variables	Constraints	Variables
544	346	3424	1210

stronger relaxations. In particular, for the case of linear GDP we showed how a hierarchy of relaxations can be developed by performing sequential intersection of disjunctions. Finally, for the case when the GDP problem involves nonconvex functions, we proposed a scheme for tightening the lower bounds for obtaining the global optimum using a combined disjunctive and spatial branch and bound search. We illustrated the application of the theoretical concepts and algorithms on several engineering and OR problems.

Acknowledgments. The authors would like to acknowledge financial support from the National Science Foundation under Grant OCI-0750826.

REFERENCES

- [1] ABHISHEK K., LEYFFER S., AND LINDEROTH J.T., *FilmINT: An Outer-Approximation-Based Solver for Nonlinear Mixed Integer Programs*, ANL/MCS-P1374-0906, Argonne National Laboratory, 2006.
- [2] BALAS E., *Disjunctive Programming*, **5**, 3–51, 1979.
- [3] BALAS E., *Disjunctive Programming and a hierarchy of relaxations for discrete optimization problems*, SIAM J. Alg. Disc. Meth., **6**, 466–486, 1985.
- [4] BEAUMONT N., *An Algorithm for Disjunctive Programs*, European Journal of Operations Research, **48**, 362–371, 1991.
- [5] BELOTTI P., LEE J., LIBERTI L., MARGOT F., AND WÄCHTER A., *Branching and bounds tightening techniques for non-convex MINLP*, Optimization Methods and Software, **24**:4, 597–634, 2009.
- [6] BENDERS J.F., *Partitioning procedures for solving mixed-variables programming problems*, Numer.Math., **4**, 238–252, 1962.
- [7] BORCHERS B. AND MITCHELL J.E., *An Improved Branch and Bound Algorithm for Mixed Integer Nonlinear Programming*, Computers and Operations Research, **21**, 359–367, 1994.
- [8] BONAMI P., BIEGLER L.T., CONN A.R., CORNUEJOLS G., GROSSMANN I.E., LAIRD C.D., LEE J. , LODI A. , MARGOT F., SAWAYA N., AND WÄCHTER A. , *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization, **5**, 186–204, 2008.
- [9] BROOKE A., KENDRICK D., MEERAUS A., AND RAMAN R., *GAMS, a User's Guide*, GAMS Development Corporation, Washington, 1998.
- [10] DURAN M.A. AND GROSSMANN I.E., *An Outer-Approximation Algorithm for a Class of Mixed-integer Nonlinear Programs*, Math Programming, **36**, p. 307, 1986.
- [11] FLETCHER R. AND LEYFFER S., *Solving Mixed Integer Nonlinear Programs by Outer-Approximation*, Math Programming, **66**, p. 327, 1994.
- [12] GALAN B. AND GROSSMANN I.E., *Optimal Design of Distributed Wastewater Treatment Networks*, Ind. Eng. Chem. Res., **37**, 4036–4048, 1998.

- [13] GEOFFRION A.M., *Generalized Benders decomposition*, JOTA, **10**, 237–260, 1972.
- [14] GROSSMANN I.E., *Review of Non-Linear Mixed Integer and Disjunctive Programming Techniques for Process Systems Engineering*, Optimization and Engineering, **3**, 227–252, 2002.
- [15] GROSSMANN I.E., CABALLERO J.A., AND YEOMANS H., *Advances in Mathematical Programming for Automated Design, Integration and Operation of Chemical Processes*, Korean J. Chem. Eng., **16**, 407–426, 1999.
- [16] GROSSMANN I.E. AND LEE S., *Generalized Convex Disjunctive Programming: Non-linear Convex Hull Relaxation*, Computational Optimization and Applications, **26**, 83–100, 2003.
- [17] GUPTA O.K. AND RAVINDRAN V., *Branch and Bound Experiments in Convex Non-linear Integer Programming*, Management Science, **31**:12, 1533–1546, 1985.
- [18] HOOKER J.N. AND OSORIO M.A., *Mixed logical-linear programming*, Discrete Applied Mathematics, **96**–**97**, 395–442, 1999.
- [19] HOOKER J.N., *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*, Wiley, 2000.
- [20] HORST R. AND TUY H., *Global Optimization deterministic approaches*, 3rd Ed, Springer-Verlag, 1996.
- [21] KALLRATH J., *Mixed Integer Optimization in the Chemical Process Industry: Experience, Potential and Future*, Trans. I .Chem E., **78**, 809–822, 2000.
- [22] LEE S. AND GROSSMANN I.E., *New Algorithms for Nonlinear Generalized Disjunctive Programming*, Computers and Chemical Engineering, **24**, 2125–2141, 2000.
- [23] LEE S. AND GROSSMANN I.E., *Global optimization of nonlinear generalized disjunctive programming with bilinear inequality constraints: application to process networks*, Computers and Chemical Engineering, **27**, 1557–1575, 2003.
- [24] LEYFFER S., *Integrating SQP and Branch and Bound for Mixed Integer Nonlinear Programming*, Computational Optimization and Applications, **18**, 295–309, 2001.
- [25] LIBERTI L., MLADENOVIC M., AND NANNICINI G., *A good recipe for solving MINLPs*, Hybridizing metaheuristics and mathematical programming, Springer, **10**, 2009.
- [26] LINDO SYSTEMS INC, *LindoGLOBAL Solver*
- [27] MENDEZ C.A., CERDA J., GROSSMANN I.E., HARJUNKOSKI I., AND FAHL M., *State-of-the-art Review of Optimization Methods for Short-Term Scheduling of Batch Processes*, Comput. Chem. Eng., **30**, p. 913, 2006.
- [28] NEMHAUSER G.L. AND WOLSEY L.A., *Integer and Combinatorial Optimization*, Wiley-Interscience, 1988.
- [29] QUESADA I. AND GROSSMANN I.E., *An LP/NLP Based Branch and Bound Algorithm for Convex MINLP Optimization Problems*, Computers and Chemical Engineering, **16**, 937–947, 1992.
- [30] QUESADA I. AND GROSSMANN I.E., *Global optimization of bilinear process networks with multicomponent flows*, Computers and Chemical Engineering, **19**:12, 1219–1242, 1995.
- [31] RAMAN R. AND GROSSMANN I.E., *Relation Between MILP Modelling and Logical Inference for Chemical Process Synthesis*, Computers and Chemical Engineering, **15**, 73, 1991.
- [32] RAMAN R. AND GROSSMANN I.E., *Modelling and Computational Techniques for Logic-Based Integer Programming*, Computers and Chemical Engineering, **18**, p. 563, 1994.
- [33] RUIZ J.P. AND GROSSMANN I.E., *Strengthening the lower bounds for bilinear and concave GDP problems*, Computers and Chemical Engineering, **34**:3, 914–930, 2010.
- [34] SAHINIDIS N.V., *BARON: A General Purpose Global Optimization Software Package*, Journal of Global Optimization, **8**:2, 201–205, 1996.

- [35] SAWAYA N. AND GROSSMANN I.E., *A cutting plane method for solving linear generalized disjunctive programming problems*, Computers and Chemical Engineering, **20**:9, 1891–1913, 2005.
- [36] SAWAYA N., *Thesis: Reformulations, relaxations and cutting planes for generalized disjunctive programming*, Carnegie Mellon University, 2006.
- [37] SCHWEIGER C.A. AND FLOUDAS C.A., *Process Synthesis, Design and Control: A Mixed Integer Optimal Control Framework*, Proceedings of DYCOPS-5 on Dynamics and Control of Process Systems, 189–194, 1998.
- [38] STUBBS R. AND MEHROTRA S. , *A Branch-and-Cut Method for 0-1 Mixed Convex Programming*, Math Programming, **86**:3, 515–532, 1999.
- [39] TAWARMALANI M. AND SAHINIDIS N., *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*, Kluwer Academic Publishers, 2002.
- [40] TURKAY M. AND GROSSMANN I.E., *A Logic-Based Outer-Approximation Algorithm for MINLP Optimization of Process Flowsheets*, Computers and Chemical Engineering, **20**, 959–978, 1996.
- [41] VECCHIETTI A., LEE S., AND GROSSMANN, I.E., *Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations*, Computers and Chemical Engineering, **27**, 433–448, 2003.
- [42] VECCHIETTI A. AND GROSSMANN I.E., *LOGMIP: A Discrete Continuous Nonlinear Optimizer*, Computers and Chemical Engineering, **23**, 555–565, 2003.
- [43] VISWANATHAN AND GROSSMANN I.E., *A combined penalty function and outer-approximation method for MINLP optimization*, Computers and Chemical Engineering, **14**, 769–782, 1990.
- [44] WESTERLUND T. AND PETTERSSON F., *A Cutting Plane Method for Solving Convex MINLP Problems*, Computers and Chemical Engineering, **19**, S131–S136, 1995.
- [45] WESTERLUND T. AND PÖRN R., *Solving Pseudo-Convex Mixed Integer Optimization Problems by Cutting Plane Techniques*, Optimization and Engineering, **3**, 253–280, 2002.
- [46] WILLIAMS H.P., *Mathematical Building in Mathematical Programming*, John Wiley, 1985.
- [47] YUAN, X., ZHANG S., PIBOLEAU L., AND DOMENECH S., *Une Methode d'optimisation Nonlineare en Variables Mixtes pour la Conception de Procedes*, RAIRO, **22**, 331, 1988.
- [48] ZAMORA J.M. AND GROSSMANN I.E., *A branch and bound algorithm for problems with concave univariate , bilinear and linear fractional terms*, **14**:3, 217–249, 1999.

DISJUNCTIVE CUTS FOR NONCONVEX MINLP

PIETRO BELOTTI*

Abstract. Mixed Integer Nonlinear Programming (MINLP) problems present two main challenges: the integrality of a subset of variables and nonconvex (nonlinear) objective function and constraints. Many exact solvers for MINLP are branch-and-bound algorithms that compute a lower bound on the optimal solution using a linear programming relaxation of the original problem.

In order to solve these problems to optimality, disjunctions can be used to partition the solution set or to obtain strong lower bounds on the optimal solution of the problem. In the MINLP context, the use of disjunctions for branching has been subject to intense research, while the practical utility of disjunctions as a means of generating valid linear inequalities has attracted some attention only recently.

We describe an application to MINLP of a well-known separation method for disjunctive cuts that has shown to be very effective in Mixed Integer Linear Programming (MILP). As the experimental results show, this application obtains encouraging results in the MINLP case even when a simple separation method is used.

Key words. Disjunctions, MINLP, Couenne, Disjunctive cuts.

AMS(MOS) subject classifications. 90C57.

1. Motivation: nonconvex MINLP. Mixed integer nonlinear programming is a powerful modeling tool for very generally defined problems in optimization [32]. A MINLP problem is defined as follows:

$$\begin{aligned} (\mathbf{P}_0) \quad & \min \quad f(x) \\ \text{s.t.} \quad & g_j(x) \leq 0 \quad \forall j = 1, 2, \dots, m \\ & \ell_i \leq x_i \leq u_i \quad \forall i = 1, 2, \dots, n \\ & x \in \mathbb{Z}^r \times \mathbb{R}^{n-r}, \end{aligned}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g_j : \mathbb{R}^n \rightarrow \mathbb{R}$, for all $j = 1, 2, \dots, m$, are in general multivariate, nonconvex functions; n is the number of variables, r is the number of integer variables, and x is the n -vector of variables, whose i -th component is denoted by x_i ; and ℓ_i and u_i are lower and upper bounds on variable x_i . We assume that all these bounds are finite, as otherwise the problem is undecidable [33].

We assume that f and all g_j 's are *factorable*, i.e., they can be computed in a finite number of simple steps, starting with model variables and real constants, using unary (e.g., log, exp, sin, cos, tan) and binary operators (e.g., $+$, $-$, $*$, $/$, $^\wedge$). Note that this framework, although very general, excludes problems whose constraints or objective function are, for example, black-box functions or indefinite integrals such as the *error function*.

There are numerous applications of \mathbf{P}_0 in Chemical Engineering [14, 26, 35] and Computational Biology [39, 40, 50], among others. Special

*Department of Mathematical Sciences, Clemson University, Clemson, SC 29634 (pbelotti@clemson.edu).

subclasses of MINLP, such as Mixed Integer Linear Programming (MILP), where f is linear and all g_i 's are affine, and convex MINLPs (i.e., MINLPs whose continuous relaxation is a convex nonlinear program), admit special (and more efficient) solvers, therefore the only reason to use a general-purpose nonconvex MINLP solver is that the problem cannot be classified as any of those special cases.

Effective algorithms for nonconvex optimization aim at finding a relaxation and obtaining a good lower bound on the optimal solution value. In the MILP case, a lower bound can be found by solving the LP relaxation obtained by relaxing integrality on the variables. In the convex MINLP case, relaxing integrality yields a convex nonlinear problem and hence a lower bound. In the general case, finding a relaxation and a lower bound on the global optimum for \mathbf{P}_0 can be hard, since relaxing integrality yields a nonconvex NLP.

Disjunctions. When the relaxation does not obtain a strong lower bound, an approach to strengthening the relaxation is to use logical *disjunctions* that are satisfied by all solutions of \mathbf{P}_0 . In their most general form, disjunctions are logical operators that return true whenever one or more of their operands are true [5]. In this work, we consider disjunctions involving linear inequalities, although more general disjunctions are possible. Let us denote as S the feasible set of \mathbf{P}_0 , i.e., $S = \{x \in \mathbb{Z}^r \times \mathbb{R}^{n-r} : \ell_i \leq x_i \leq u_i \forall i = 1, 2, \dots, n, g_j(x) \leq 0 \forall j = 1, 2, \dots, m\}$. A disjunction is an operator on a set of systems of inequalities over the set S , written as

$$\bigcup_{h \in Q} \{x \in \mathbb{R}^n : A^h x \leq b^h, x \in S\}, \quad (1.1)$$

where $A^h \in \mathbb{Q}^{m_h \times n}$, $b^h \in \mathbb{Q}^{m_h}$, m_h is the number of inequalities in the h -th system, and Q is an index set. We say that the disjunction is true if there exist $x \in S$ and $h \in Q$ such that $A^h x \leq b^h$.

The general definition (1.1) comprises several classes of disjunctions. Two important ones are the *integer disjunction* $x_i \leq \alpha \vee x_i \geq \alpha + 1$, which is valid for any $\alpha \in \mathbb{Z}$ if x_i is an integer variable; and the *spatial disjunction* $x_i \leq \beta \vee x_i \geq \beta$, with $\beta \in \mathbb{R}$ and x_i a continuous variable.

If disjunctions are used in the branching step or to generate disjunctive cuts, two main problems need to be addressed: *(i)* how to describe the set of disjunctions and *(ii)* what disjunction should be used among the (possibly infinite) set of valid disjunctions.

Branch-and-bound algorithms. The *branch* operation partitions the feasible set of an optimization problem by imposing a disjunction. Each term of a disjunction (1.1) is associated with a subproblem \mathbf{P}_h of the form

$$\begin{aligned}
(\mathbf{P}_h) \quad & \min f(x) \\
\text{s.t. } & g_j(x) \leq 0 \quad \forall j = 1, 2, \dots, m \\
& A^h x \leq b^h \\
& \ell_i \leq x_i \leq u_i \quad \forall i = 1, 2, \dots, n \\
& x \in \mathbb{Z}^r \times \mathbb{R}^{n-r},
\end{aligned} \tag{1.2}$$

in which the constraints (1.2) are those imposed by branching. We denote the feasible region of \mathbf{P}_h by S_h .

A *branch-and-bound* (BB) algorithm applies this branching method recursively. Any MINLP with a bounded feasible region can be solved by a BB in a finite number of steps [32]. Note that an optimal solution of a subproblem \mathbf{P}_h is not necessarily optimal for \mathbf{P}_0 , since the feasible regions of individual subproblems are subsets of the original feasible region and may not include an optimal solution of \mathbf{P}_0 . Nevertheless, solutions obtained in this way are still feasible and hence yield valid upper bounds on the optimal value of $f(x)$. An advantage of applying a branching method is that it allows to discard any subproblem whose associated lower bound exceeds the best upper bound found so far.

Disjunctive cuts. Disjunctions can also be imposed *indirectly* by deriving one or more implied constraints from them [5]. A valid inequality with respect to S consists of a pair $(\alpha, \alpha_0) \in \mathbb{Q}^{n+1}$ such that $\alpha^\top x \leq \alpha_0$ for all $x \in S$. An inequality that is valid for S and violated by at least one member of the feasible region of the relaxation of \mathbf{P}_0 is called a *cut*.

A *disjunctive cut* with respect to a disjunction of the form (1.1) is an inequality that is valid for S_h for each $h \in Q$, or equivalently, for the convex hull of the union of these sets. Amending disjunctive cuts to the feasible region of a relaxation of \mathbf{P}_0 is often an effective way of tightening a relaxation of S .

Disjunctive cuts have long been studied in the MILP context [7, 9, 49]. They have also been generated from disjunctions arising from MINLPs with a certain structure. For MINLP with binary variables and whose continuous relaxation is convex, Stubbs and Mehrotra [60] generalized the procedure proposed by Balas, Ceria and Cornuéjols [7] and described a separation procedure based on a convex optimization problem.

A specialized procedure has been successfully applied to Mixed Integer Conic Programming (MICP) problems, which are MILPs amended by a set of constraints whose feasible region is a cone in \mathbb{R}^n . The second order cone and the cone of symmetric semidefinite matrices are among the most important classes of conic constraints in this class. Çezik and Iyengar [19] and lately Drewes [25] proposed, for MICP problems where all disjunctions are generated from binary variables, an application of the procedure to the conic case, where disjunctive inequalities are obtained by solving a continuous conic optimization problem. Analogously, Frangioni and Gentile [27]

describe a class of disjunctive cuts that can be used in convex MINLPs where binary variables are used to model *semi-continuous* variables, i.e., variables that take values in the set $\{0\} \cup [l, u]$ with $0 \notin [l, u]$.

Another interesting class of problems is that of Mathematical Programs with Complementarity Constraints (MPCC) [57], i.e., MINLPs that contain nonconvex constraints $x^\top y = 0$, with $x \in \mathbb{R}_+^k$, $y \in \mathbb{R}_+^k$. These can be more easily stated as $x_i y_i = 0$ for all $i = 1, 2, \dots, k$, and give rise to simple disjunctions $x_i = 0 \vee y_i = 0$. Júdice et al. [34] study an MPCC where the only nonlinear constraints are the complementarity constraints, and therefore relaxing the latter yields an LP. Disjunctive cuts are generated from solutions of the LP that violate a complementarity constraint (i.e., solutions where $x_i > 0$ and $y_i > 0$ for some $i \in \{1, 2, \dots, k\}$) through the observation that both variables are basic and by applying standard disjunctive arguments to the corresponding tableau rows.

Scope of the paper and outline. While the application to MILP of cuts derived from disjunctions has been studied for decades and efficient implementations are available, their practical application to MINLP has recently attracted a lot of attention: a notable example is the work by Saxena et al. [55, 56], where disjunctive cuts are used to solve MINLPs with quadratic nonconvex objective and constraints.

In particular, [55] point out that “... even though the results presented in this paper focussed on MIQCPs, they are equally applicable to a much wider class of nonconvex MINLPs. All we need is an automatic system that can take a nonconvex MINLP and extract a corresponding MIQCP relaxation. Development of software such as Couenne [11, 12] is a step in this direction.”

Our contribution is an attempt to follow that indication: we present an application of disjunctive programming to the context of nonconvex MINLP problems. We provide an Open Source implementation of a procedure that generates disjunctive cuts for MINLP problems, and present a set of computational results that substantiate the practical utility of this application.

Several exact solution methods for nonconvex MINLP rely, in order to obtain valid lower bounds, on reformulation and linearization techniques [46, 59, 61], which we briefly introduce in the next section as their definition is essential to describe the type of disjunctions we use in this context. Section 3 describes the general disjunctions arising from reformulating an MINLP, and their use in separating disjunctive cuts is shown in Section 4.

A simple separation procedure based on the cut generating LP (CGLP) is discussed in Section 5. This procedure has been added to COUENNE, a general-purpose, Open Source solver for MINLP [11], and tested on a set of publicly available nonconvex MINLP instances. These tests are presented in Section 6.

2. Lower bounds of an MINLP. Several MINLP solvers are BB algorithms whose lower bounding technique uses a Linear Programming (LP) relaxation constructed in two steps:

- *reformulation*: \mathbf{P}_0 is transformed in an equivalent MINLP with a set of new variables, a linear objective function, and a set of nonlinear equality constraints;
- *linearization*: each nonlinear equality constraint is relaxed by replacing it with a set of linear inequalities.

Here we provide a brief description whose purpose is to introduce the derivation of disjunctions in Section 3.

Reformulation. If f and all g_i 's are factorable, they can be represented by *expression trees*, i.e., n -ary trees where every node is either a constant, a variable, or an n -ary operator whose arguments are represented as children of the node, and which are in turn expression trees [21, Chapter 3]. While all leaves of an expression tree are variables or constants, each non-leaf node, which is an operator of a finite set $\mathcal{O} = \{+, *, ^\wedge, /, \sin, \cos, \exp, \log\}$, is associated with a new variable, denoted as *auxiliary variable*. As a result, we obtain a *reformulation* of \mathbf{P}_0 [46, 59, 61]:

$$\begin{aligned} (\mathbf{P}) \quad \min \quad & x_{n+q} \\ \text{s.t.} \quad & x_k = \vartheta_k(x) \quad k = n+1, n+2, \dots, n+q \\ & \ell_i \leq x_i \leq u_i \quad i = 1, 2, \dots, n+q \\ & x \in X \end{aligned}$$

where $X = \mathbb{Z}^r \times \mathbb{R}^{n-r} \times \mathbb{Z}^s \times \mathbb{R}^{q-s}$, that is, q auxiliary variables are introduced, s of which are integral and $q-s$ continuous¹. The auxiliary x_{n+q} is associated to the operator that defines the objective function of \mathbf{P}_0 . Each auxiliary x_k is associated with a function $\vartheta_k(x)$ from \mathcal{O} and depends, in general, on one or more of the variables x_1, x_2, \dots, x_{k-1} . Let us define the bounding box $[\ell, u]$ of \mathbf{P} as the set $\{x \in \mathbb{R}^{n+q} : \ell_i \leq x_i \leq u_i, i = 1, 2, \dots, n+q\}$.

In the reformulation, all nonlinear constraints are of the form $x_k = \vartheta_k(x)$ for all $k = n+1, n+2, \dots, n+q$, where ϑ_k is an operator from \mathcal{O} . It is then possible to obtain an LP relaxation of \mathbf{P} , or *linearization* for short, by applying operator-specific algorithms to each such constraint [59].

Linearization. Consider a nonlinear constraint $x_k = \vartheta_k(x)$ in the reformulation \mathbf{P} . The set $\Theta_k = \{x \in X \cap [\ell, u] : x_k = \vartheta_k(x)\}$ is, in general, nonconvex. The nonconvex set of feasible solutions of the reformulation is therefore $\Theta = \bigcap_{k=n+1}^{n+q} \Theta_k$.

While finding a convex relaxation of Θ might prove very difficult, obtaining a convex relaxation of each Θ_k , for all $k = n+1, n+2, \dots, n+q$, is

¹The integrality and the bounds on x_k , for $k = n+1, n+2, \dots, n+q$, are inferred from the associated function ϑ_k and the bounds and the integrality of its arguments.

in general easier and has been extensively studied [46, 59, 61, 41]. Several exact MINLP solvers [54, 12, 42] seek an LP relaxation of Θ_k defined by the set $\mathbf{LP}_k = \{x \in [\ell, u] : B^k x \leq c^k\}$, with $B^k \in \mathbb{Q}^{m_k \times (n+q)}$, $c^k \in \mathbb{Q}^{m_k}$, and m_k is the number of inequalities of the relaxation. In general, m_k depends on ϑ_k and, for certain constraints, a larger m_k yields a better lower bound. However, in order to keep the size of the linearization limited, a relatively small value is used. For the linearization procedure we have used, $m_k \leq 4$ for all k . Let us denote $\mathbf{LP} = \bigcap_{k=n+1}^{n+q} \mathbf{LP}_k$. Because $\mathbf{LP}_k \supseteq \Theta_k$ for all $k = n+1, n+2, \dots, n+q$, we have $\mathbf{LP} \supseteq \Theta$. Hence, minimizing x_{n+q} over the convex set \mathbf{LP} gives a lower bound on the optimal solution value of \mathbf{P}_0 .

The aforementioned MINLP solvers are branch-and-bound procedures that obtain, at every BB node, a linearization of \mathbf{P} . It is crucial, as pointed out by McCormick [46], that for each $k = n+1, n+2, \dots, n+q$, the linearization of Θ_k be *exact* at the lower and upper bounds on the variables appearing as arguments of ϑ_k , i.e., the linearization must be such that if a solution x^* is feasible for the LP relaxation but not for \mathbf{P} , then there exists an i such that $\ell_i < x_i^* < u_i$, so that a spatial disjunction $x_i \leq x_i^* \vee x_i \geq x_i^*$ can be used.

In general, for all $k = n+1, n+2, \dots, n+q$, both B^k and c^k depend on the variable bounds ℓ and u : the tighter the variable bounds, the stronger the lower bound obtained by minimizing x_{n+q} over \mathbf{LP} . For such an approach to work effectively, several *bound reduction* techniques have been developed [30, 48, 52, 51, 58, 38]. Most of these techniques use the nonconvex constraints of the reformulation or the linear constraints of the linearization, or a combination of both. An experimental comparison of bound reduction techniques used in a MINLP solver is given in [12].

Let us consider, as an example, a constraint $x_k = \vartheta_k(x) = (x_i)^3$, and the nonconvex set $\Theta_k = \{x \in \mathbb{R}^{n+q} \cap [\ell, u] : x_k = (x_i)^3\}$, whose projection on the (x_i, x_k) space is the bold curve in [Figure 1\(a\)](#). A linearization of Θ_k is in [Figure 1\(b\)](#), and is obtained through a procedure based on the function $\vartheta_k(x) = (x_i)^3$ and the bounds on x_i . As shown in [Fig. 1\(c\)](#), when tighter bounds are known for x_i a better relaxation can be obtained: the set $\Theta'_k = \{x \in \mathbb{R}^{n+q} \cap [\ell, u'] : x_k = (x_i)^3\}$, where $u'_j = u_j$ for $j \neq i$ and $u'_i < u_i$, admits a tighter linearization \mathbf{LP}' , which is a proper subset of \mathbf{LP} since the linearization is exact at u'_i .

3. Disjunctions in MINLP. In order to apply the paradigm of disjunctive programming to nonconvex MINLP, we need (*i*) a description of the set of valid disjunctions in \mathbf{P} and (*ii*) a procedure for selecting, from said set, a disjunction violated by an optimal solution x^* to the LP relaxation.

Deriving disjunctions. Nonconvex constraints in the reformulation \mathbf{P} above are of two types:

- integrality of a subset of variables: x_i for $i \in \{1, 2, \dots, r, n+1, n+1, \dots, n+s\}$;

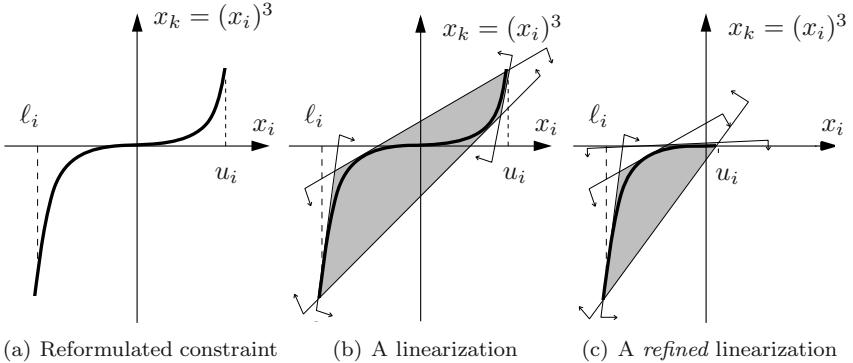


FIG. 1. Linearization of constraint $x_k = (x_i)^3$: the bold line in (a) represents the nonconvex set $\{(x_i, x_k) : \ell_i \leq x_i \leq u_i, x_k = (x_i)^3\}$, while the polyhedra in (b) and (c) are its linearizations for different bounds on x_i .

- nonconvex constraints $x_k = \vartheta_k(x)$ for $k = n+1, n+2, \dots, n+q$.

MILP problems and convex MINLPs contain nonconvex constraints of the first type only. When these constraints are relaxed, one obtains a continuous relaxation which yields a lower bound on the optimal solution value and a solution vector x^* that satisfies the convex constraints, but may violate the integrality requirements. If $x_i^* \notin \mathbb{Z}$ for one of the integer variables, then x^* violates the *variable disjunction* $x_i \leq \lfloor x_i^* \rfloor \vee x_i \geq \lceil x_i^* \rceil$.

There has been extensive research on how to use this type of disjunction, how to derive disjunctive cuts from them, and how to efficiently add such inequalities, using a BB method coupled with a cut generating procedure [7, 8, 49]. Several generalizations can be introduced at the MILP level: a well known example is the *split disjunction* $\pi x \leq \pi_0 \vee \pi x \geq \pi_0 + 1$, where $(\pi, \pi_0) \in \mathbb{Z}^{p+1}$ and $x \in \mathbb{Z}^p$ is a vector of p integer variables [10, 17, 22, 36].

The reformulation of an MINLP problem is subject to both types of nonconvex constraints. The optimal solution to the LP relaxation, x^* , may be infeasible for the integrality constraints or for one or more of the constraints of the reformulation, $x_k = \vartheta_k(x)$. In this work, we will primarily consider disjunctions arising from constraints of the second type, and will focus on the problem of finding a valid disjunction that is violated by a solution to the LP relaxation of \mathbf{P}_0 .

For the sake of simplicity, consider a nonconvex constraint $x_k = \vartheta_k(x_i)$ with ϑ_k univariate and x_i continuous (disjunctions can be derived with a similar procedure when ϑ_k is multivariate and/or x_i is integer), and assume this constraint is violated by a solution x^* of the LP relaxation, i.e., $x_k^* \neq \vartheta_k(x_i^*)$. The *spatial disjunction*

$$x_i \leq \beta \quad \vee \quad x_i \geq \beta, \tag{3.1}$$

although valid for any $\beta \in [\ell_i, u_i]$, is *not* violated by x^* . This is a point of departure with MILP, for which disjunctions over integer variables are valid and, obviously, violated by fractional solutions.

A violated disjunction for MINLP, however, can be derived by applying bound reduction and the linearization procedure. Suppose that the linearization for the set $\Theta'_k = \{x \in X \cap [\ell, u] : x_i \leq \beta, x_k = \vartheta_k(x_i)\}$ is $\mathbf{LP}'_k = \{x \in [\ell, u] : B'x \leq c'\}$, and analogously, the linearization for $\Theta''_k = \{x \in X \cap [\ell, u] : x_i \geq \beta, x_k = \vartheta_k(x_i)\}$ is $\mathbf{LP}''_k = \{x \in [\ell, u] : B''x \leq c''\}$; we assume that the linear constraints include the new upper (resp. lower) bound β on x_i . Assuming that the two linearizations \mathbf{LP}'_k and \mathbf{LP}''_k are *exact* at the bounds on x_i , we have $x^* \notin \mathbf{LP}'_k \cup \mathbf{LP}''_k$. Hence, the disjunction

$$x \in \mathbf{LP}'_k \quad \vee \quad x \in \mathbf{LP}''_k \quad (3.2)$$

is valid and violated by x^* , and can be used to generate a disjunctive cut or a branching rule.

Although it might seem intuitive to set $\beta = x_i^*$, in practice spatial disjunctions (3.1) are often selected such that $\beta \neq x_i^*$, for instance because x_i^* is very close to one of the bounds on x_i . Even so, there are procedures to derive a valid disjunction (3.2) violated by x^* . The rules for selecting a good value of β have been discussed, among others, in [12, 62] and will not be presented here.

As an example, consider the constraint $x_k = \vartheta_k(x_i) = (x_i)^2$ and the components (x_i^*, x_k^*) of the LP solution x^* as in Figure 2(a). Since x^* lies in the convex hull of $\Theta_k = \{x \in \mathbb{R}^{n+q}, x \in [\ell, u], x_k = (x_i)^2\}$, refining the linearization, by adding linearization inequalities for other auxiliary variables, may not be sufficient to separate x^* . In this case, a valid disjunction that is violated by x^* can be used to either create two subproblems or to derive a disjunctive cut. We observe that (x_i^*, x_k^*) is not necessarily a vertex of the linearization of $x_k = (x_i)^2$ since it is simply a projection of x^* (which is a vertex of the LP relaxation) onto the plane (x_i, x_k) . Starting from the valid disjunction $x_i \leq \beta \vee x_i \geq \beta$, one can create two linearizations \mathbf{LP}' and \mathbf{LP}'' , shown as shaded areas in Figure 2(b). The corresponding disjunction $x \in \mathbf{LP}' \vee x \in \mathbf{LP}''$ is valid and violated by x^* .

Selecting a disjunction. In its simplest form, the problem of finding the most effective disjunction considers variable disjunctions only and can be stated as follows: given an optimal solution x^* of an LP relaxation of \mathbf{P} , an index set I of integer variables with fractional value in x^* , and an index set N of violated nonconvex constraints of \mathbf{P} , i.e., $N = \{k \in \{n+1, n+1, \dots, n+q\} : x_k^* \neq \vartheta_k(x^*)\}$, choose either (i) an element $i \in I$ defining the integer disjunction $x_i \leq \lfloor x_i^* \rfloor \vee x_i \geq \lceil x_i^* \rceil$ or (ii) an element k of N , a variable x_i that is an argument of $\vartheta_k(x)$, and a scalar β defining the spatial disjunction $x_i \leq \beta \vee x_i \geq \beta$.

In order to select a disjunction, an *infeasibility* parameter, which depends on the solution x^* of the LP relaxation, is associated with each variable x_i of the reformulation. A null infeasibility corresponds to a disjunction satisfied by x^* , and a nonzero infeasibility corresponds to a disjunction violated by x^* . Ideally, a large infeasibility should indicate a better disjunction, for example in terms of improvement in the lower bound of the two problems generated by branching on that disjunction. The infeasibility $\Omega_{\text{int}}(x_i)$ of an integer variable x_i is computed as $\min\{x_i^* - \lfloor x_i^* \rfloor, \lceil x_i^* \rceil - x_i^*\}$. For a variable x_j appearing in one or more nonlinear expressions, consider the set D_j of indices k of nonconvex constraints $x_k = \vartheta_k(x)$ where ϑ_k contains x_j as an argument, and define $\gamma_k = |x_k^* - \vartheta_k(x^*)|$ for all k in D_j . We define the infeasibility $\Omega_{\text{nl}}(x_i)$ of x_j as a convex combination of $\min_{k \in D_j} \gamma_k$, $\sum_{k \in D_j} \gamma_k$, and $\max_{k \in D_j} \gamma_k$. If x_j is also integer, the maximum between $\Omega_{\text{nl}}(x_i)$ and $\Omega_{\text{int}}(x_i)$ is used. We refer the reader to [12, §5] for a more detailed description, which is omitted here for the sake of conciseness.

A standard procedure for selecting disjunctions sorts them in non-increasing order of infeasibility. When selecting a disjunction for branching, the one with maximum infeasibility is chosen. If a set of disjunctive cuts is desired, the first p disjunctions in the sorted list are chosen, for a given $p \geq 1$, and p disjunctive cuts are generated.

As discussed in [1, 12], this infeasibility measure may not be the best way to rank disjunctions. More sophisticated techniques have been proposed for disjunction selection in the branching process, for example strong branching [4], pseudocosts branching [13], reliability branching [1], and Violation Transfer [43, 62]. A generalization of reliability branching [1] to the MINLP case has been recently presented [12].

Disjunctions in special cases of MINLP. Some classes of MINLP problems exhibit nonconvex constraints that give rise to special disjunctions, and are used to generate disjunctive cuts. One such example has been studied for quadratically constrained quadratic programs (QCQPs) by Saxena et al. [55, 56]. These problems contain nonlinear terms of the form $x_i x_j$. Applying a reformulation such as that described in Section 2 would obtain auxiliary variables $y_{ij} = x_i x_j$ for $1 \leq i \leq j \leq n$. In matricial form, this corresponds to the nonconvex constraint $Y = xx^\top$, where Y is a symmetric, $n \times n$ matrix of auxiliary variables and x is the n -vector of variables (notice that there are $n(n+1)/2$ new variables instead of n^2 , since $y_{ij} = y_{ji}$). Rather than applying a linearization to each nonconvex constraint $y_{ij} = x_i x_j$, such a constraint can be relaxed as $Y - xx^\top \succeq 0$, thus reducing a QCQP to a (convex) Semidefinite program, which yields comparably good lower bounds [31, 53, 3]. However, these SDP models are obtained from the original problem by relaxing the nonconvex constraint $xx^\top - Y \succeq 0$. In [55], this constraint is used to generate disjunctions: given a vector $v \in \mathbb{R}^n$, the non-convex constraint $(v^\top x)^2 \geq v^\top Y v$, which can be rewritten as $w^2 \geq z$ after a change of variables, is obtained from the neg-

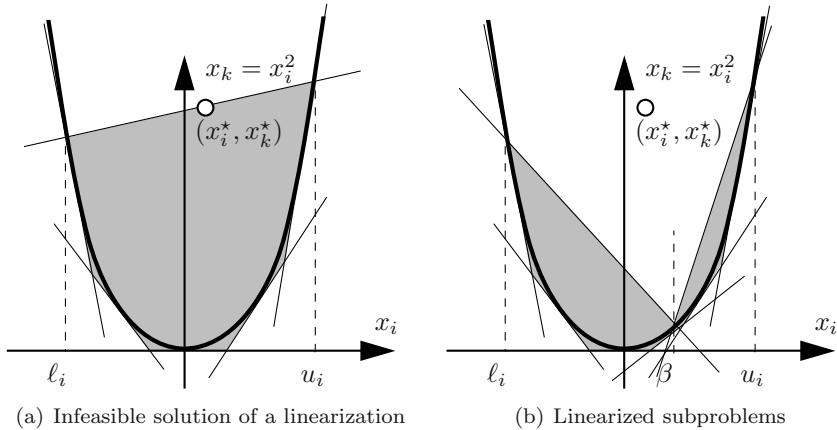


FIG. 2. *MINLP Disjunctions and nonconvex constraints.* In (a), the shaded area is the linearization of the constraint $x_k = \vartheta_k(x_i)$ with $x_i \in [\ell_i, u_i]$, whereas (x_i^*, x_k^*) is the value of x_i and x_k in the optimum of the LP relaxation. In (b), the spatial disjunction $x_i \leq \beta \vee x_i \geq \beta$ generates two sets $\Theta'_k = \{x \in X \cap [\ell, u] : x_k = x_i^2, x_i \leq \beta\}$ and $\Theta''_k = \{x \in X \cap [\ell, u] : x_k = x_i^2, x_i \geq \beta\}$. The corresponding linearizations, \mathbf{LP}'_k and \mathbf{LP}''_k , are the smaller shaded areas.

ative eigenvalues of the matrix $\bar{x}\bar{x}^\top - \bar{Y}$, where (\bar{x}, \bar{Y}) is a solution to the relaxation. This is then used to generate a disjunction and disjunctive cuts in x and Y . In a more recent development [56], this procedure is modified to generate disjunctive cuts in the original variables x only.

4. Disjunctive cuts in nonconvex MINLP. Assume that the linearization step produces an LP relaxation that we denote $\min\{x_{n+q} : Ax \leq a, \ell \leq x \leq u\}$, where $A \in \mathbb{Q}^{K \times (n+q)}$, $a \in \mathbb{Q}^K$, and K is the total number of linear inequalities generated for all of the nonlinear constraints $x_k = \vartheta_k(x)$ of the reformulation (we recall that $K \leq 4(n+q)$), while ℓ and u are the vectors of the lower and upper bounds on both original and auxiliary variables. Let us denote the feasible set of the linear relaxation as $\mathbf{LP} = \{x \in \mathbb{R}^{n+q} : Ax \leq a, \ell \leq x \leq u\}$.

Consider a disjunction $x_i \leq \beta \vee x_i \geq \beta$. The two LP relaxations

$$\begin{aligned}\mathbf{LP}' &= \{x \in \mathbf{LP} : x_i \leq \beta\} \\ \mathbf{LP}'' &= \{x \in \mathbf{LP} : x_i \geq \beta\},\end{aligned}$$

are created by amending to \mathbf{LP} one constraint of the disjunction. As pointed out in the previous section, the disjunction alone does not eliminate any solution from $\mathbf{LP}' \cup \mathbf{LP}''$, because $\mathbf{LP} = \mathbf{LP}' \cup \mathbf{LP}''$, while this does not hold for disjunctions on integer variables, where \mathbf{LP} strictly contains the union of the two subproblems. Consider two problems \mathbf{P}' and \mathbf{P}'' , obtained by intersecting \mathbf{P} with the constraints $x_i \leq \beta$ and $x_i \geq \beta$, respectively. Apply bound reduction and the linearization procedure to \mathbf{P}'

and \mathbf{P}'' , and denote the tightened problems as \mathbf{SLP}' and \mathbf{SLP}'' (see Figure 2(b)). The two sets can be described as

$$\begin{aligned}\mathbf{SLP}' &= \{x \in \mathbf{LP} : B'x \leq c'\}, \\ \mathbf{SLP}'' &= \{x \in \mathbf{LP} : B''x \leq c''\},\end{aligned}$$

where $B' \in \mathbb{Q}^{H' \times (n+q)}$, $c' \in \mathbb{Q}^{H'}$, $B'' \in \mathbb{Q}^{H'' \times (n+q)}$, and $c'' \in \mathbb{Q}^{H''}$ are the coefficient matrices and the right-hand side vectors of the inequalities added to the linearizations, which contain the new bound on variable x_i and, possibly, new bounds on other variables.

We re-write these two sets in a more convenient form:

$$\begin{aligned}\mathbf{SLP}' &= \{x \in \mathbb{R}^{n+q} : A'x \leq a'\} \\ \mathbf{SLP}'' &= \{x \in \mathbb{R}^{n+q} : A''x \leq a''\},\end{aligned}$$

where

$$A' = \begin{pmatrix} A \\ B' \\ -I \\ I \end{pmatrix}, \quad a' = \begin{pmatrix} a \\ c' \\ -\ell \\ u \end{pmatrix}; \quad A'' = \begin{pmatrix} A \\ B'' \\ -I \\ I \end{pmatrix}, \quad a'' = \begin{pmatrix} a \\ c'' \\ -\ell \\ u \end{pmatrix}$$

so as to include the initial linear constraints, the new linearizations inequalities, and the variable bounds in a more compact notation. We denote as K' (resp. K'') the number of rows of A' (resp. A'') and the number of elements of a' (resp. a'').

As described by Balas [5], an inequality $\alpha^\top x \leq \alpha_0$, where $\alpha \in \mathbb{Q}^{n+q}$ and $\alpha_0 \in \mathbb{Q}$, is valid for the convex hull of $\mathbf{SLP}' \cup \mathbf{SLP}''$ if α and α_0 satisfy:

$$\begin{aligned}\alpha &\leq u^\top A', \quad \alpha_0 = u^\top a', \\ \alpha &\leq v^\top A'', \quad \alpha_0 = v^\top a'',\end{aligned}$$

where $u \in \mathbb{R}_+^{K'}$ and $v \in \mathbb{R}_+^{K''}$. Given an LP relaxation and its optimal solution x^* , an automatic procedure for generating a cut of this type consists of finding vectors u and v such that the corresponding cut is maximally violated [7]. This requires solving the Cut Generating Linear Programming (CGLP) problem

$$\begin{array}{lll}\max & \alpha^\top x^* & -\alpha_0 \\ \text{s.t.} & \alpha & -u^\top A' \\ & \alpha & -v^\top A'' \\ & \alpha_0 & -u^\top a' \\ & \alpha_0 & -v^\top a'' \\ & u^\top e & +v^\top e \\ & u, v & \geq 0\end{array} \leq 0 \quad (4.1)$$

where e is the vector with all components equal to one. An optimal solution (more precisely, any solution with positive objective value) provides a

valid disjunctive cut that is violated by the current solution x^* . Its main disadvantage is its size: the CGLP has $(n + q + 1) + K' + K''$ variables and $2(n + q) + 3$ constraints, and is hence at least twice as large as the LP used to compute a lower bound. Given that the optimal solution of the CGLP is used, in our implementation, to produce just one disjunctive cut, solving one problem (4.1) for each disjunction of a set of violated ones, at every branch-and-bound node, might prove ineffective. To this purpose, Balas et al. [6, 9] present a method to implicitly solve the CGLP for binary disjunctions by applying pivot operations to the original linear relaxation, only with a different choice of variables. It is worth noting that, unlike the MILP case, here A' and A'' differ for much more than a single column. As shown in [2], this implies that the result by Balas et al. does not hold in this case.

An example. Consider the continuous nonconvex nonlinear program $\mathbf{P}_0 : \min\{x^2 : x^4 \geq 1\}$. It is immediate to check that its feasible region is the nonconvex union of intervals $(-\infty, -1] \cup [+1, +\infty)$, and that its two global minima are -1 and $+1$. Its reformulation is as follows:

$$\begin{aligned} (\mathbf{P}) \quad & \min \quad w \\ \text{s.t.} \quad & w = x^2 \\ & y = x^4 \\ & y \geq 1. \end{aligned}$$

It is crucial to note here that, although the problem is trivial and can be solved by inspection, state-of-the-art MINLP solvers that use reformulation *ignore* the relationship between the objective function and the constraint, i.e., $y = w^2$. The tightest convex relaxations of the two nonlinear constraints are obtained by simply replacing the equality with inequality, therefore any LP relaxation generated by a MINLP solver is a relaxation of

$$\begin{aligned} (\mathbf{CR}) \quad & \min \quad w \\ \text{s.t.} \quad & w \geq x^2 \\ & y \geq x^4 \\ & y \geq 1, \end{aligned}$$

whose optimal solution is $(x, w, y) = (0, 0, 1)$, whose value is $w = 0$ and which is infeasible for \mathbf{P} and hence for \mathbf{P}_0 . Imposing the disjunction $x \leq 0 \vee x \geq 0$ to \mathbf{P} yields two subproblems with the following convex relaxations:

$$\begin{aligned} (\mathbf{CR}') \quad & \min\{w : y \geq 1, w \geq x^2, y \geq x^4, x \leq 0\} \\ (\mathbf{CR}'') \quad & \min\{w : y \geq 1, w \geq x^2, y \geq x^4, x \geq 0\}, \end{aligned}$$

both with the same optimal solution, $(x, w, y) = (0, 0, 1)$. Bound reduction is crucial here for both subproblems, as it strengthens the bounds on x using the lower bound on y . Indeed, $x \leq 0$ and $1 \leq y = x^2$ imply $x \leq -1$,

and analogously $x \geq 0$ and $1 \leq y = x^2$ imply $x \geq 1$. Hence, the relaxations of the two tightened subproblems are

$$\begin{aligned} (\mathbf{SCR}') & \min\{w : y \geq 1, w \geq x^2, y \geq x^4, x \leq -1\} \\ (\mathbf{SCR}'') & \min\{w : y \geq 1, w \geq x^2, y \geq x^4, x \geq +1\} \end{aligned}$$

and their optimal solutions are feasible for \mathbf{P}_0 and correspond to the two global optima $\{-1, +1\}$. Hence, the problem is solved after branching on the disjunction $x \leq 0 \vee x \geq 0$. However, the nonlinear inequality $x^2 \geq 1$ is valid for both \mathbf{CR}' and \mathbf{CR}'' as it is implied by $x \leq -1$ in the former and by $x \geq 1$ in the latter. Since $w \geq x^2$, the (linear) disjunctive cut $w \geq 1$ is valid for both (\mathbf{SCR}') and (\mathbf{SCR}'') . If added to (\mathbf{CR}) , a lower bound of 1 is obtained which allows to solve the problem without branching. It is easy to prove that even using a linear relaxation and applying the CGLP procedure yields the same disjunctive cut. This simple example can be complicated by considering n variables subject each to a nonconvex constraint:

$$\begin{array}{ll} \min & \sum_{i=1}^n x_i^2 \\ \text{s.t.} & x_i^4 \geq 1 \quad \forall i = 1, 2, \dots, n. \end{array}$$

A standard BB implementation needs to branch on all variables before closing the gap between lower and upper bound, requiring an exponential number of subproblems as one needs to branch on all disjunctions. However, the disjunctive cuts $w_i \geq 1 \forall i = 1, 2, \dots, n$, where w_i is the variable associated with the expression x_i^2 , allow to find an optimal solution immediately. Although this example uses a nonlinear convex hull for the problem and the subproblems, it can be shown that the same disjunctive cut can be generated within a framework where linear relaxations are used and a CGLP-based method for generating disjunctive cuts is used.

A test with $n = 100$ was conducted using the two variants of COUENNE DC and v0 described in Section 6, respectively with and without disjunctive cuts. With disjunctive cuts generated at all nodes (at most 20 per node), the problem was solved to global optimality (the optimal solution has a value of 100) in 57 seconds and 18 BB nodes on a Pentium M 2.0GHz laptop, 41 seconds of which were spent in the generation of such cuts. Without disjunctive cuts, the solver was stopped after two hours of computation, more than 319,000 active BB nodes, and a lower bound of 14.

As a side note, one might expect a BB procedure to enforce all disjunctions as branching rules. Hence, at any node of depth d in the BB tree the d disjunctions applied down to that level determine the bound to be equal to d . Because all optima have objective value equal to n , an exponential number of nodes will have to be explored, so that the global lower bound of the BB will increase as the logarithm of the number of nodes.

This example can be interpreted as follows: the decomposition of the expression trees at the reformulation phase is followed by a linearization that only takes into account, for each nonlinear constraint $y_i = x_i^4$, of the

variables x_i and y_i only — this causes a bad lower bound as there is no link between the lower bound on y_i and that on w_i . The disjunctive cuts, while still based on a poor-quality LP relaxation, have a more “global” perspective of the problem, where the bound on y_i implies a bound on w_i .

5. A procedure to generate disjunctive cuts. The procedure is sketched in [Table 1](#), and its details are discussed below. It requires a description of \mathbf{P} , the feasible region of its linear relaxation $\mathbf{LP} = \{x \in \mathbb{R}^{n+q} : Ax \leq a\}$, lower and upper bounds ℓ, u , an optimal solution x^* of the relaxation, and disjunction $x_i \leq \beta \vee x_i \geq \beta$. Branch-and-bound procedures for MINLP recursively reduce the bounding box $[\ell, u]$, therefore the procedure sketched can be applied at every branch-and-bound node.

Implementation details. We have used COUENNE [11, 23], an Open Source software package included in the Coin-OR infrastructure [44], for all experiments. It implements reformulation, linearization, and bound reduction methods, as well as a *reliability branching* scheme [12]. It also recognizes complementarity constraints, i.e., constraints of the form $x_i x_j = 0$, with $i \neq j$, as the disjunction $x_i = 0 \vee x_j = 0$.

COUENNE is a BB whose lower bounding method is based on the reformulation and linearization steps as discussed in Section 2. At the beginning, the linearization procedure is run in order to obtain an initial LP relaxation. At each node of the BB, two cut generation procedures are used: up to four rounds of cuts to refine the linear relaxation and at most one round of disjunctive cuts.

Calls to the disjunctive cut generator. While the procedure that generates a linear relaxation of the MINLP is relatively fast and hence is carried out multiple times at every node of the BB tree, separating disjunctive cuts using a CGLP is CPU-intensive as it requires solving a relatively large LP for each disjunction. Therefore, in our experiments, disjunctive cuts are only generated at BB nodes of depth lesser than 10, and at most 20 violated disjunctions per node are used.

For each node where disjunctive cuts are separated, the first 20 disjunctions of a list provided by the branch-and-bound algorithm (which, in COUENNE’s case, is CBC [18]) are selected. This list is sorted in non-increasing order of the infeasibility of variable x_i of the reformulation and a solution x^* of the LP relaxation.

Disjunctions and convex/concave functions. Disjunctions can help cut an LP solution through branching or disjunctive cuts, but they are not the only way to do so. In order to generate disjunctive cuts only when necessary, in our experiments a disjunction from a nonlinear constraint $x_k = \vartheta_k(x)$ is *not* used, for branching or for separating a disjunctive cut, if it is possible to add a linearization inequality for ϑ_k that cuts x^* , i.e., if $x_k^* > \vartheta_k(x^*)$ and ϑ_k is concave or $x_k^* < \vartheta_k(x^*)$ and ϑ_k is convex. Consider [Figure 3](#), where the components (x_i^*, x_k^*) of the LP solution associated with the nonlinear constraint $x_k = e^{x_i}$ are shown in two cases: in the

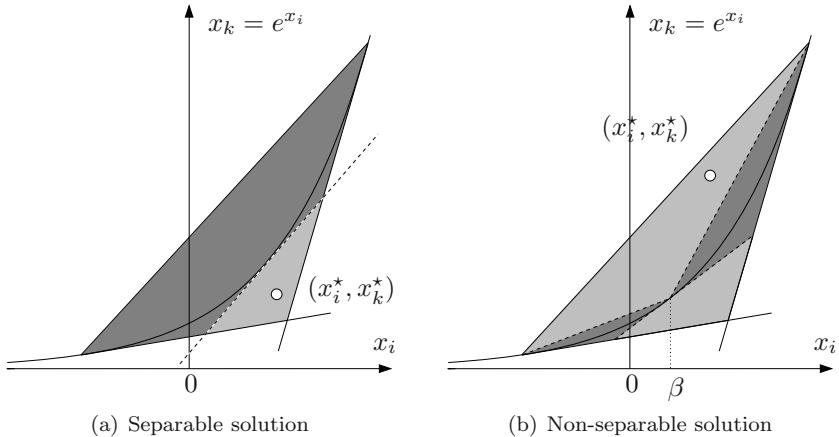


FIG. 3. *Separable and non-separable points.* In (a), although the point is infeasible for \mathbf{P} , another round of linearization cuts is preferred to a disjunction (either by branching or through a disjunctive cut), as it is much quicker to separate. In (b), no refinement of the linear relaxation is possible, and a disjunction must be applied.

first, $x_k^* < e^{x_i^*}$, while in the second $x_k^* > e^{x_i^*}$. In both cases, a disjunction $x_i \leq \beta \vee x_i \geq \beta$ could be considered by the MINLP solver as a means to create two new subproblems and the two LP relaxations, both excluding x^* , or to create a disjunctive cut. However, the point x^* in Figure 3(a) can be cut simply by refining the LP relaxation, thus avoiding the evaluation of a disjunction on x_i .

Disjunctions on integer variables. Disjunctions of the form $x_i \leq \lfloor x_i^* \rfloor \vee x_i \geq \lceil x_i^* \rceil$ are also used in this framework, as they may also lead to two subproblems with refined relaxation and tightened bounds. A more efficient separation technique for generating disjunctive cuts based on integer disjunctions, CGLLANDP [20], is available in the COIN-OR cut generation library [45]. COUENNE has an option for separating cuts using CGLLANDP, but can also use the procedure described above on integer disjunctions. The two procedures are different: while the first uses a variant of the method proposed by Balas et al. [6, 9] and is hence faster, the CGLP-based procedure in COUENNE takes advantage of the reduced bounds and the refined linearization, which are not available to CGLLANDP. In our experiments, we have turned off CGLLANDP and separated disjunctive cuts on integer variables with the method described in this paper, with the only difference that the disjunction used is of the form $x_i \leq \beta \vee x_i \geq \beta + 1$, with $\beta \in \mathbb{Z}$.

Branching priorities. In the software framework we chose for our implementation, integer and nonlinear disjunctions have a *priority*, i.e., a scalar that determines precedences in choosing disjunctions: a disjunction can be selected only if there are no violated disjunctions with smaller priority.

TABLE 1
Procedure for generating a disjunctive cut for problem \mathbf{P} .

Input:	A problem \mathbf{P} and a linear relaxation, \mathbf{LP} : (A, a, ℓ, u) Optimal solution of \mathbf{LP} : x^* A disjunction $x_i \leq \beta \vee x_i \geq \beta$
1	Create \mathbf{P}' from \mathbf{P} by imposing $x_i \leq \beta$
2	Create \mathbf{P}'' from \mathbf{P} by imposing $x_i \geq \beta$
3	Apply bound reduction to \mathbf{P}' , obtain $[\ell', u']$
4	Apply bound reduction to \mathbf{P}'' , obtain $[\ell'', u'']$
5	Generate linear relaxations \mathbf{SLP}' of \mathbf{P}' , defined by A', a'
6	Generate linear relaxations \mathbf{SLP}'' of \mathbf{P}'' , defined by A'', a''
7	Construct CGLP (4.1) and solve it
8	Return (α, α_0)

In our experiments, we have assigned the same priority to all disjunctions, therefore disjunctions are chosen based on their infeasibility only.

Rank of the generated inequalities. Suppose an LP relaxation \mathbf{LP} and K disjunctions are available at a branch-and-bound node. In our implementation, after generating the disjunctive cut $\alpha^\top x \leq \alpha_0$ for the j -th disjunction, the cut for the $(j+1)$ -st disjunction is generated using a CGLP constructed from the same LP relaxation \mathbf{LP} , i.e., *without* the new cut $\alpha^\top x \leq \alpha_0$ (COUENNE has an option to amend the CGLP with the cuts generated in the same round, which was not tested). This is done to limit the rank of the new cut, given that high-rank cuts may introduce numerical errors. In fact, the K -th generated cut would otherwise be of rank K even if \mathbf{LP} contained no disjunctive cuts, and, in the worst case, its rank would be Kd if generated in a branch-and-bound node at depth d . Therefore, given the maximum node depth of 10 mentioned above, the maximum rank of the cuts separated in the tests below is 10. If, in the future, disjunctive cuts are generated at deeper nodes of the BB tree, mechanisms for controlling the rank of each cut, such as that used by [55], might be necessary.

6. Experimental results. In order to assess the utility and effectiveness of the procedure to generate disjunctive cuts for MINLP described in this paper, we have performed a battery of tests on a set of 84 publicly available MINLP instances from the following repositories:

- *MacMINLP* [37] and *minplib* [16]: a collection of MINLP instances, both convex and nonconvex;
- *nConv*: a collection of nonconvex MINLPs²;
- *MIQQP*: Mixed Integer quadratically constrained quadratic programs [47]; model *qpsi.mod* was used;
- *globallib*: a collection of continuous NLP problems [29];

²See <https://projects.coin-or.org/Couenne/browser/problems/nConv>.

- *boxQP*: continuous, nonconvex, box-constrained quadratic problems; the smaller instances are from [63] and those with more than 60 variables are from [15];
- *airCond*, a 2D bin-packing problem for air conduct design [28].

While most of these instances are nonconvex MINLP problems, the *boxQP* instances belong to a specific class of problems for which much more efficient methods exist. In particular, the one presented by Burer et al. [15] solves all these instances more quickly than COUENNE. It should also be noted that, for this type of problems, there are much stronger relaxations than that obtained through reformulation, see for example Anstreicher [3].

Table 3 describes the parameters of each instance: number of variables (*var*), of integer variables (*ivar*), of constraints (*con*), and of auxiliary variables (*aux*), or, in the notation used above: *n*, *r*, *m*, and *q*. The latter parameter is a good indicator of the size of the LP relaxation, as it is proportional to the number of linearization inequalities added.

We have conducted tests to compare two distinct branching techniques with disjunctive cuts, in order to understand what combination is most effective. The following four variants of COUENNE have been tested:

- v0: no disjunctive cuts, and the basic branching scheme **br-plain** described in [12], Section 5.1;
- RB: no disjunctive cuts, and an extension of reliability branching [1] to MINLP denoted **int-br-rev** in [12];
- DC: disjunctive cuts separated until depth 10 of the BB tree and the **br-plain** branching scheme;
- DC+RB: disjunctive cuts separated until depth 10 of the BB tree and reliability branching.

The latter variant, apart from being a combination of more sophisticated methods, has a further advantage: since reliability branching is a method to rank branching rules and therefore disjunctions, disjunctive cuts are separated only using the most promising disjunctions.

All tests were performed on a computer with a 2.66GHz processor, 64GB of RAM, and Linux kernel 2.6.29. COUENNE version 0.2, compiled with *gcc 4.4.0*, was used. A time limit of two hours was set for all variants.

Tables 4 and **5** report in detail the comparison between the four variants of COUENNE. If an algorithm solved an instance in less than two hours, its CPU time in seconds is reported. Otherwise, the *remaining gap*

$$\text{gap} = \frac{z_{\text{best}} - z_{\text{lower}}}{z_{\text{best}} - z_0} \quad (6.1)$$

is given, where z_{best} is the objective value of the best solution found by the four algorithms, z_{lower} is the lower bound obtained by this algorithm, and z_0 is the initial lower bound found by COUENNE, which is the same for all variants. If no feasible solution was found by any variant, the lower bound is reported in brackets. The best performances are highlighted in bold.

TABLE 2

Summary of the comparison between the four variants. The first three columns report, for those instances that could be solved within the time limit of two hours, the number of instances solved (“solved”), the number of instances for which the variant obtained the best CPU time or within the 10% of the best (“best time”), and analogously for the number of BB nodes (“best nodes”). The last column, “best gap,” reports the number of instances, among those that could not be solved within two hours by any of the variant, for which a variant obtained the smallest gap, or within 10% of the smallest, in terms of the remaining gap.

Variant	solved	<2h best time	best nodes	Unsolved best gap
v0	26	12	11	31
RB	26	13	11	35
DC	35	8	13	29
DC+RB	39	20	29	37

Table 2 summarizes our results by pointing out what variants perform better overall. For each variant, the column “solved” reports the number of instances solved by that variant before the time limit, while column “best time” reports the number of solved instances whose CPU time is best among the four variants, or at most 10% greater than the best time. An analogous measure is given in the third column, “best nodes,” for the BB nodes. The last column, “best gap,” refers to instances that were not solved before the time limit by any of the variants, and reports for how many of these the variant had the best gap, or within 10% of the best.

Although this gives a somewhat limited perspective, it shows that the variants with disjunctive cuts, especially the one coupled with reliability branching, have an edge over the remaining variants. They in fact allow to solve more problems within the time limit, on average, and, even when a problem is too difficult to solve, the remaining gap is smaller more often when using disjunctive cuts.

Variants with disjunctive cuts seem to outperform the others, especially for the *boxQP* instances (we recall that more efficient solvers are available for this type of problem [15]). For some instances, however, disjunctive cuts only seem to slow down the BB as the CPU time spent in separation does not produce any effective cut. The tradeoff between the effectiveness of the disjunctive cuts and the time spent generating them suggests that a faster cut generation would increase the advantage.

We further emphasize this fact by showing the amount of time spent by reliability branching and disjunctive cuts, which is included in the total CPU time reported. **Tables 6** and **7** show, for a subset of instances for which branching time or separation time were relatively large (at least 500 seconds), the CPU time spent in both processes and the number of resulting cuts or BB nodes. This selection of instances shows that, in certain cases, the benefit of disjunctive cuts is worth the CPU time spent

in the generation. This holds especially for the *boxQP* instances: although a large amount of time is spent in generating disjunctive cuts, this results in a better lower bound or a lower CPU time. Again, the fact that the current separation algorithm is rather simple suggests that a more efficient implementation would obtain the same benefit in shorter time.

We also graphically represent the performance of the four variants using performance profiles [24]. [Figure 4\(a\)](#) depicts a comparison on the CPU time. This performance profile only considers instances that could be solved in less than two hours by at least one of the variants. Hence, it also compares the quality of a variant in terms of number of instances solved. [Figure 4\(b\)](#) is a performance profile on the number of nodes.

[Figure 4\(c\)](#) is a comparison on the remaining gap, and reports on all instances for which *none* of the variants could obtain an optimal solution in two hours or less. Note that this is not a performance profile: rather than the *ratio* between gaps, this graph shows, for each algorithm, the number of instances (plotted on the *y* axis) with remaining gap below the corresponding entry on the *x* axis.

The three graphs show once again that, for the set of instances we have considered, using both reliability branching and disjunctive cuts pay off for both easy and difficult MINLP instances. The former are solved in shorter time, while for the latter we yield a better lower bound.

7. Concluding remarks. Disjunctive cuts are as effective in MINLP solvers as they are in MILP. Although they are generated from an LP relaxation of a nonconvex MINLP, they can dramatically improve the lower bound and hence the performance of a branch-and-bound method.

One disadvantage of the CGLP procedure, namely having to solve a large LP in order to obtain a single cut, carries over to the MINLP case. Some algorithms have been developed, for the MILP case, to overcome this issue [6, 9]. Unfortunately, as shown in [2], their extension to the MINLP case is not as straightforward.

Acknowledgments. The author warmly thanks François Margot for all the useful discussions that led to the development of this work, and an anonymous referee whose feedback helped improve this article. Part of this research was conducted while the author was a Postdoctoral Fellow at the Tepper School of Business, Carnegie Mellon University, Pittsburgh PA.

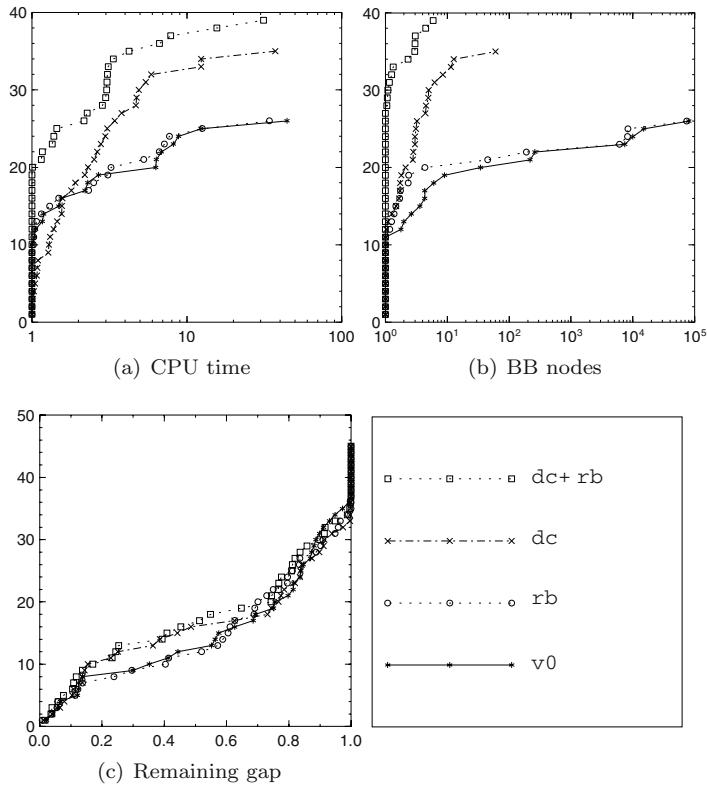


FIG. 4. *Performance profiles for the four variants of COUENNE.*

TABLE 3
Instances used in our tests. For each instance, “var” is the number of integer variables, “con” the number of constraints, and “aux” the number of auxiliary variables generated at the reformulation step. Instances in the boxQP group are continuous and only constrained by a bounding box; hence columns “ivar” and “con” are omitted.

TABLE 4
Comparison between the four methods. Each entry is either the CPU time, in seconds, taken to solve the instance or, if greater than two hours, the remaining gap (6.1) after two hours. If the remaining gap cannot be computed due to the lack of a feasible solution, the lower bound, in brackets, is shown instead.

Name	v0	RB	boxQP	DC	DC+RB	Name	v0	RB	globalib	DC	DC+RB	
sp020-100-1	57	70	21	9	9	catmix100	9	10	14	13	13	
sp030-060-1	1925	1864	503	280	lits100	30.0%	29.6%	15.4%	25.4%	25.4%	25.4%	
sp030-070-1	3.4%	2.2%	1.129	718	camsh200	79.7%	61.1%	78.5%	64.7%	64.7%	64.7%	
sp030-080-1	13.7%	10.7%	1406	1342	qp2	12.5%	11.3%	13.8%	10.5%	10.5%	10.5%	
sp030-090-1	2591	1662	695	316	catmix200	6.6%	5.5%	6.5%	5.9%	5.9%	5.9%	
sp030-100-1	12.9%	11.3%	2169	1654	turkey	53	57	57	53	53	53	
sp040-030-1	74	60	9	8	qp1	11.4	127	110	127	127	127	
sp040-040-1	3.1%	5.2%	393	308	elec50	12.4%	12.2%	14.4%	11.8%	11.8%	11.8%	
sp040-050-1	4.5%	8.2%	703	370	camsh400	84.7%	72.9%	84.1%	(353.6)	(353.6)	(353.6)	
sp040-060-1	39.4%	36.3%	6467	4145	ark0002	(0)	(0)	(0)	(0)	(0)	(0)	
sp040-070-1	27.5%	24.5%	2746	1090	polygon50	(-20.2)	(-20.2)	(-20.2)	(-20.2)	(-20.2)	(-20.2)	
sp040-080-1	39.6%	40.4%	6.4%	6389	ark0019	13.7%	13.7%	13.7%	13.7%	13.7%	13.7%	
sp040-090-1	41.2%	41.4%	7.9%	7.6%	ark0015	83.6%	83.6%	83.6%	83.6%	83.6%	83.6%	
sp040-100-1	44.3%	40.4%	10.4%	7.6	infeas1	62.6%	62.6%	62.5%	54.9%	54.9%	54.9%	
sp050-030-1	35.4	36.1	31	29	lits400	5.7%	5.7%	10.8%	10.8%	10.8%	10.8%	
sp050-040-1	29.6%	28.5%	1669	632	camsh800	87.3%	95.9%	97.4%	95.0%	95.0%	95.0%	
sp050-050-1	57.5%	57.2%	22.8%	17.0%	ark0010	1622	1538	2126	2079	2079	2079	
sp060-020-1	1.241	953	36	28	<i>nConv</i>							
sp070-025-1	40.1%	40.8%	2292	824	<i>c-sched47</i>							
sp070-050-1	69.3%	70.1%	36.4%	45.3%	<i>syntheadmod</i>							
sp070-075-1	81.4%	79.5%	76.7%	76.8%	<i>JoseneN5c</i>							
sp080-025-1	53.4%	49.5%	47115	1241	<i>MIQQP</i>							
sp080-050-1	81.6%	81.2%	74.4%	74.4%	<i>ivalues</i>							
sp080-075-1	86.9%	88.7%	82.0%	81.2%	<i>imis07</i>							
sp090-025-1	68.6%	69.1%	38.6%	24.5%	<i>ibcl</i>							
sp090-050-1	83.9%	83.4%	77.4%	77.6%	<i>iswath2</i>							
sp090-075-1	94.8%	94.9%	87.4%	91.7%	<i>imas84</i>							
sp100-025-1	75.6%	75.0%	48.6%	40.9%	<i>ieild76</i>							
sp100-050-1	89.8%	90.9%	83.7%	82.0%	<i>globalib</i>							
sp100-075-1	97.3%	96.6%	91.4%	91.4%	<i>DC+RB</i>							

TABLE 5

Comparison between the four methods. Each entry is either the CPU time, in seconds, taken to solve the instance or, if greater than two hours, the remaining gap (6.1) after two hours. If the remaining gap cannot be computed due to the lack of a feasible solution, the lower bound, in brackets, is shown instead.

Name	V0	RB	DC	DC+RB
<i>minlplib</i>				
waterz	75.1%	58.7%	73.2%	85.9%
ravem	16	23	74	66
ravempb	18	24	55	42
enpro56	39	26	156	76
enpro56pb	55	24	301	81
csched2	2.2%	3.9%	1.2%	3.6%
water4	55.2%	52.0%	44.2%	51.3%
enpro48	58	37	204	114
enpro48pb	49	41	201	126
space25a	(116.4)	(107.2)	(107.3)	(100.1)
contvar	91.1%	90.2%	91.1%	89.9%
space25	(98.0)	(96.7)	(97.1)	(177.6)
lop97icx	93.0%	79.6%	93.9%	39.3%
du-opt5	73	170	251	159
du-opt	87	270	136	262
waste	(255.4)	(439.3)	(273.7)	(281.6)
lop97ic	(2543.5)	(2532.8)	(2539.4)	(2556.7)
qapw	(0)	(20482)	(0)	(20482)
<i>MacMINLP</i>				
trimlon4	23	4	133	24
trimlon5	48.1%	46	35.6%	142
trimlon6	(16.17)	(18.69)	(16.18)	(18.52)
trimlon7	88.1%	60.5%	89.8%	74.3%
space-25-r	(74.2)	(68.6)	(75.0)	(69.6)
space-25	(98.4)	(89.6)	(96.3)	(91.9)
trimlon12	(16.1)	(18.6)	(16.1)	(18.5)
space-960-i	(6.5e+6)	(6.5e+6)	(6.5e+6)	(6.5e+6)
<i>misc.</i>				
airCond	187	0.9%	876	1471

TABLE 6

Comparison of time spent in the separation of disjunctive cuts (t_{sep}) and in reliability branching (t_{br}). Also reported is the number of nodes (“nodes”) and of disjunctive cuts generated (“cuts”).

Name	RB		DC		DC+RB			
	t_{br}	nodes	t_{sep}	cuts	t_{sep}	cuts	t_{br}	nodes
<i>boxQP</i>								
sp040-060-1	8	340k	3104	17939	2286	11781	56	3k
sp040-070-1	10	277k	1510	5494	466	2042	56	277k
sp040-080-1	14	174k	4030	15289	3813	14831	95	4k
sp040-090-1	58	136k	4142	13733	3936	14604	115	3k
sp040-100-1	17	178k	4317	11415	3882	11959	126	2k
sp050-050-1	12	289k	4603	10842	4412	11114	127	6k
sp070-025-1	38	308k	1324	2215	383	921	28	308k
sp070-050-1	38	80k	5027	3823	4616	3622	477	80k
sp070-075-1	81	26k	6125	1220	5951	1089	46	26k
sp080-025-1	27	222k	2873	2818	716	1154	28	222k
sp080-050-1	66	35k	5888	1864	5561	1740	85	35k
sp080-075-1	119	4k	6449	720	6464	615	32	4k
sp090-025-1	32	170k	4672	3540	4386	3044	500	170k
sp090-050-1	86	26k	6064	1023	6335	838	37	26k
sp090-075-1	194	4k	6468	467	6862	250	8	4k
sp100-025-1	54	80k	5286	2990	4704	2372	659	80k
sp100-050-1	140	4k	6376	693	6363	733	34	4k
sp100-075-1	272	35k	6852	312	6835	302	3	35k
<i>globallib</i>								
lnts100	6084	4k	3234	16	1195	6	4647	3k
camsh200	6242	10k	1772	2303	2123	2645	4192	9k
qp2	214	62k	2662	3505	1351	1662	444	41k
qp3	1298	803k	43	737	47	644	3272	484k
qp1	189	63k	3160	3525	1609	1736	240	45k
elec50	5677	45k	5494	45	5159	68	768	45k
camsh400	3494	33k	3433	818	5242	1263	560	53k
arki0002	6834	53k	3571	237	1641	181	5277	53k
arki0019	5761	53k	1846	39	1543	36	3812	53k
arki0015	2442	2k	2141	215	1442	106	2412	2k
infeas1	1306	2k	1495	45	1273	131	1397	2k
lnts400	457	2k	4767	1	4724	0	373	2k
camsh800	5001	23k	3671	188	2208	100	3172	23k

TABLE 7

(Continued) Comparison of time spent in the separation of disjunctive cuts (t_{sep}) and in reliability branching (t_{br}). Also reported is the number of nodes (“nodes”) and of disjunctive cuts generated (“cuts”).

Name	RB		DC		DC+RB			
	t_{br}	nodes	t_{sep}	cuts	t_{sep}	cuts	t_{br}	nodes
<i>nConv</i>								
JoseSEN5c	5166	1061k	4280	39	341	5	5315	1061k
<i>MIQP</i>								
ivalue	2816	10k	4223	700	4589	733	1571	10k
imisc07	6005	2k	6835	2621	5475	2131	1404	2k
ibc1	310	2k	6166	38	6368	167	46	2k
iswath2	827	2k	6567	31	6526	52	5	2k
imas284	443	62k	6769	1408	4474	700	381	72k
ieilD76	2934	9k	6994	75	6869	56	804	9k
<i>minplib</i>								
contvar	4284	2k	11	19	13	21	4706	3k
space25	272	1051k	2133	706	178	309	311	1100k
lop97icx	6540	14k	1142	353	4226	2603	2649	3k
waste	5204	13k	7090	715	6934	253	82	13k
lop97ic	3178	11k	4556	25	6734	147	3547	11k
qapw	6400	61k	104	0	34	0	6367	60k
<i>MacMINLP</i>								
trimlon6	8500	62k	58	50	558	2649	5944	63k
trimlon12	8432	62k	58	50	562	2649	5941	62k
space-960-i	5859	62k	2127	20	1624	0	4971	62k
<i>misc.</i>								
airCond	7123	112k	293	1736	103	1526	39	541k

REFERENCES

- [1] T. ACHTERBERG, T. KOCH, AND A. MARTIN, *Branching rules revisited*, OR Letters, **33** (2005), pp. 42–54.
- [2] K. ANDERSEN, G. CORNUÉJOLS, AND Y. LI, *Split closure and intersection cuts*, Mathematical Programming, **102** (2005), pp. 457–493.
- [3] K.M. ANSTREICHER, *Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming*, Journal of Global Optimization, **43** (2009), pp. 471–484.
- [4] D.L. APPLEGATE, R.E. BIXBY, V. CHVÁTAL, AND W.J. COOK, *The Traveling Salesman Problem, A Computational Study*, Princeton, 2006.
- [5] E. BALAS, *Disjunctive programming: Properties of the convex hull of feasible points*, Discrete Applied Mathematics, **89** (1998), pp. 3–44.
- [6] E. BALAS AND P. BONAMI, *New variants of Lift-and-Project cut generation from the LP tableau: Open Source implementation and testing*, in Integer Programming and Combinatorial Optimization, Vol. **4513** of Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2007, pp. 89–103.
- [7] E. BALAS, S. CERIA, AND G. CORNUÉJOLS, *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, Mathematical Programming, **58** (1993), pp. 295–324.
- [8] ———, *Mixed 0-1 programming by lift-and-project in a branch-and-cut framework*, Management Science, **42** (1996), pp. 1229–1246.
- [9] E. BALAS AND M. PERREGAARD, *A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming*, Mathematical Programming, **94** (2003), pp. 221–245.
- [10] E. BALAS AND A. SAXENA, *Optimizing over the split closure*, Mathematical Programming, Series A, **113** (2008), pp. 219–240.
- [11] P. BELOTTI, COUENNE: *a user's manual*, tech. rep., Lehigh University, 2009.
- [12] P. BELOTTI, J. LEE, L. LIBERTI, F. MARGOT, AND A. WÄCHTER, *Branching and bounds tightening techniques for non-convex minlp*, Optimization Methods and Software, **24** (2009), pp. 597–634.
- [13] M. BENICHOU, J.M. GAUTHIER, P. GIRODET, G. HENTGES, G. RIBIERE, AND O. VINCENT, *Experiments in mixed-integer linear programming*, Mathematical Programming, **1** (1971), pp. 76–94.
- [14] L.T. BIEGLER, I.E. GROSSMANN, AND A.W. WESTERBERG, *Systematic Methods of Chemical Process Design*, Prentice Hall, Upper Saddle River (NJ), 1997.
- [15] S. BURER AND D. VANDENBUSSCHE, *Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound*, Comput. Optim. Appl., **43** (2009), pp. 181–195.
- [16] M.R. BUSSIECK, A.S. DRUD, AND A. MEERAUS, *MINLPLib – a collection of test models for mixed-integer nonlinear programming*, INFORMS Journal of Computing, **15** (2003), pp. 114–119. Available online at <http://www.gamsworld.org/minlp/minplib/minlpstat.htm>.
- [17] A. CAPRARO AND A.N. LETCHFORD, *On the separation of split cuts and related inequalities*, Mathematical Programming, Series B, **94** (2003), pp. 279–294.
- [18] CBC-2.1, Available from <https://projects.coin-or.org/cbc>.
- [19] M.T. ÇEZIK AND G. IYENGAR, *Cuts for Mixed 0-1 Conic Programming*, Mathematical Programming, Ser. A, **104** (2005), pp. 179–200.
- [20] CGLLANDP, <https://projects.coin-or.org/Cgl/wiki/CglLandP>.
- [21] J.S. COHEN, *Computer algebra and symbolic computation: elementary algorithms*, A.K. Peters, Ltd., 2002.
- [22] W. COOK, R. KANNAN, AND A. SCHRIJVER, *Chvátal closures for mixed integer programming problems*, Mathematical Programming, **47** (1990), pp. 155–174.
- [23] COUENNE, <https://www.coin-or.org/Couenne>.
- [24] E.D. DOLAN AND J.J. MORÉ, *Benchmarking optimization software with performance profiles*, Mathematical Programming, **91** (2002), pp. 201–213.

- [25] S. DREWES, *Mixed Integer Second Order Cone Programming*, PhD thesis, Technische Universität Darmstadt, 2009.
- [26] C.A. FLOUDAS, *Global optimization in design and control of chemical process systems*, Journal of Process Control, **10** (2001), pp. 125–134.
- [27] A. FRANGIONI AND C. GENTILE, *Perspective cuts for a class of convex 0-1 mixed integer programs*, Mathematical Programming, **106** (2006), pp. 225–236.
- [28] A. FÜGENSCHUH AND L. SCHEWE, *Solving a nonlinear mixed-integer sheet metal design problem with linear approximations*, work in progress.
- [29] GAMS DEVELOPMENT CORP., *Gamsworld global optimization library*. <http://www.gamsworld.org/global/globallib/globalstat.htm>.
- [30] E. HANSEN, *Global Optimization Using Interval Analysis*, Marcel Dekker, Inc., New York, 1992.
- [31] C. HELMBERG AND F. RENDL, *Solving quadratic (0,1)-problems by semidefinite programs and cutting planes*, Mathematical Programming, **82** (1998), pp. 291–315.
- [32] R. HORST AND H. TUY, *Global Optimization: Deterministic Approaches*, Springer Verlag, Berlin, 1996.
- [33] R.C. JEROSLOW, *There cannot be any algorithm for integer programming with quadratic constraints*, Operations Research, **21** (1973), pp. 221–224.
- [34] J.J. JÚDICE, H.D. SHERALI, I.M. RIBEIRO, AND A.M. FAUSTINO, *A complementarity-based partitioning and disjunctive cut algorithm for mathematical programming problems with equilibrium constraints*, Journal of Global Optimization, **36** (2006), pp. 89–114.
- [35] J. KALLRATH, *Solving planning and design problems in the process industry using mixed integer and global optimization*, Annals of Operations Research, **140** (2005), pp. 339–373.
- [36] M. KARAMANOV AND G. CORNUÉJOLS, *Branching on general disjunctions*, Mathematical Programming (2007).
- [37] S. LEYFFER, MacMINLP: AMPL collection of MINLPs. Available at <http://www-unix.mcs.anl.gov/~leyffer/MacMINLP>.
- [38] L. LIBERTI, *Writing global optimization software*, in Global Optimization: from Theory to Implementation, L. Liberti and N. Maculan, eds., Springer, Berlin, 2006, pp. 211–262.
- [39] L. LIBERTI, C. LAVOR, AND N. MACULAN, *A branch-and-prune algorithm for the molecular distance geometry problem*, International Transactions in Operational Research, **15** (2008), pp. 1–17.
- [40] L. LIBERTI, C. LAVOR, M.A.C. NASCIMENTO, AND N. MACULAN, *Reformulation in mathematical programming: an application to quantum chemistry*, Discrete Applied Mathematics, **157** (2009), pp. 1309–1318.
- [41] L. LIBERTI AND C.C. PANTELIDES, *Convex envelopes of monomials of odd degree*, Journal of Global Optimization, **25** (2003), pp. 157–168.
- [42] LINDO SYSTEMS, *Lindo solver suite*. Available online at <http://www.gams.com/solvers/lindoglobal.pdf>.
- [43] M.L. LIU, N.V. SAHINIDIS, AND J.P. SHECTMAN, *Planning of chemical process networks via global concave minimization*, in Global Optimization in Engineering Design, I. Grossmann, ed., Springer, Boston, 1996, pp. 195–230.
- [44] R. LOUGEE-HEIMER, *The Common Optimization INterface for Operations Research*, IBM Journal of Research and Development, **47** (2004), pp. 57–66.
- [45] R. LOUGEE-HEIMER, *Cut generation library*. <http://projects.coin-or.org/Cgl>, 2006.
- [46] G.P. MCCORMICK, *Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems*, Mathematical Programming, **10** (1976), pp. 146–175.
- [47] H. MITTELMANN, *A collection of mixed integer quadratically constrained quadratic programs*. http://plato.asu.edu/ftp/ampl_files/miqp_ampl.

- [48] R.E. MOORE, *Methods and Applications of Interval Analysis*, Siam, Philadelphia, 1979.
- [49] J.H. OWEN AND S. MEHROTRA, *A disjunctive cutting plane procedure for general mixed-integer linear programs*, Mathematical Programming, **89** (2001), pp. 437–448.
- [50] A.T. PHILLIPS AND J.B. ROSEN, *A quadratic assignment formulation of the molecular conformation problem*, tech. rep., CSD, Univ. of Minnesota, 1998.
- [51] I. QUESADA AND I.E. GROSSMANN, *Global optimization of bilinear process networks and multicomponent flows*, Computers & Chemical Engineering, **19** (1995), pp. 1219–1242.
- [52] H. RATSCHEK AND J. ROKNE, *Interval methods*, in Handbook of Global Optimization, R. Horst and P. M. Pardalos, eds., Vol. **1**, Kluwer Academic Publishers, Dordrecht, 1995, pp. 751–828.
- [53] F. RENDL AND R. SOTIROV, *Bounds for the quadratic assignment problem using the bundle method*, Mathematical Programming, **109** (2007), pp. 505–524.
- [54] N.V. SAHINIDIS, BARON: *A general purpose global optimization software package*, Journal of Global Optimization, **8** (1996), pp. 201–205.
- [55] A. SAXENA, P. BONAMI, AND J. LEE, *Disjunctive cuts for non-convex mixed integer quadratically constrained programs*, in Proceedings of the 13th Integer Programming and Combinatorial Optimization Conference, A. Lodi, A. Panconesi, and G. Rinaldi, eds., Vol. **5035** of Lecture Notes in Computer Science, 2008, pp. 17–33.
- [56] ———, *Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations*, Mathematical Programming (2011). To appear.
- [57] H. SCHEEL AND S. SCHOLTES, *Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity*, Mathematics of Operations Research, **25** (2000), pp. 1–22.
- [58] E.M.B. SMITH, *On the Optimal Design of Continuous Processes*, PhD thesis, Imperial College of Science, Technology and Medicine, University of London, Oct. 1996.
- [59] E.M.B. SMITH AND C.C. PANTELIDES, *A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs*, Computers & Chem. Eng., **23** (1999), pp. 457–478.
- [60] R.A. STUBBS AND S. MEHROTRA, *A branch-and-cut method for 0-1 mixed convex programming*, Mathematical Programming, **86** (1999), pp. 515–532.
- [61] M. TAWARMALANI AND N.V. SAHINIDIS, *Convexification and global optimization in continuous and mixed-integer nonlinear programming: Theory, algorithms, software and applications*, Vol. **65** of Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, Dordrecht, 2002.
- [62] ———, *Global optimization of mixed-integer nonlinear programs: A theoretical and computational study*, Mathematical Programming, **99** (2004), pp. 563–591.
- [63] D. VANDENBUSSCHE AND G.L. NEMHAUSER, *A branch-and-cut algorithm for non-convex quadratic programs with box constraints*, Mathematical Programming, **102** (2005), pp. 559–575.

PART III:

NONLINEAR PROGRAMMING

SEQUENTIAL QUADRATIC PROGRAMMING METHODS

PHILIP E. GILL* AND ELIZABETH WONG*

Abstract. In his 1963 PhD thesis, Wilson proposed the first sequential quadratic programming (SQP) method for the solution of constrained nonlinear optimization problems. In the intervening 48 years, SQP methods have evolved into a powerful and effective class of methods for a wide range of optimization problems. We review some of the most prominent developments in SQP methods since 1963 and discuss the relationship of SQP methods to other popular methods, including augmented Lagrangian methods and interior methods.

Given the scope and utility of nonlinear optimization, it is not surprising that SQP methods are still a subject of active research. Recent developments in methods for mixed-integer nonlinear programming (MINLP) and the minimization of functions subject to differential equation constraints has led to a heightened interest in methods that may be “warm started” from a good approximate solution. We discuss the role of SQP methods in these contexts.

Key words. Large-scale nonlinear programming, SQP methods, nonconvex programming, quadratic programming, KKT systems.

AMS(MOS) subject classifications. 49J20, 49J15, 49M37, 49D37, 65F05, 65K05, 90C30.

1. Introduction. This paper concerns the formulation of methods for solving the smooth nonlinear programs that arise as subproblems within a method for mixed-integer nonlinear programming (MINLP). In general, this subproblem has both linear and nonlinear constraints, and may be written in the form

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} && f(x) \\ & \text{subject to} && \ell \leq \left\{ \begin{array}{l} x \\ Ax \\ c(x) \end{array} \right\} \leq u, \end{aligned} \tag{1.1}$$

where $f(x)$ is a linear or nonlinear objective function, $c(x)$ is a vector of m nonlinear constraint functions $c_i(x)$, A is a matrix, and ℓ and u are vectors of lower and upper bounds. Throughout, we assume that the number of variables is large, and that A and the derivatives of f and c are sparse. The constraints involving the matrix A and functions $c_i(x)$ will be called the *general* constraints; the remaining constraints will be called *bounds*. We assume that the nonlinear functions are smooth and that their first and second derivatives are available. An *equality* constraint corresponds to

*Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112 (pgill@ucsd.edu, elwong@ucsd.edu). This material is based upon work supported by the National Science Foundation under Grant No. DMS-0915220, and the Department of Energy under grant DE-SC0002349.

setting $\ell_i = u_i$. Similarly, a special “infinite” value for ℓ_i or u_i is used to indicate the absence of one of the bounds.

The nonlinear programs that arise in the context of MINLP have several important features. First, the problem is one of a sequence of many related NLP problems with the same objective but with additional linear or nonlinear constraints. For efficiency, it is important that information from the solution of one problem be used to “warm start” the solution of the next. Second, many MINLP methods generate infeasible subproblems as part of the solution process (e.g., in branch and bound methods). This implies that infeasible constraints will occur for a significant subset of problems to be solved, which is in contrast to the situation in conventional optimization, where constraint infeasibility is considered to be a relatively unusual event, caused by an error in the constraint formulation. In mixed-integer linear programming, the phase 1 simplex method provides a reliable “certificate of infeasibility”, i.e., a definitive statement on whether or not a feasible point exists. However, the question of the solvability of a set of nonconvex inequalities is NP-hard. Many optimization methods replace the solvability problem by a minimization problem in which the norm of the constraint residual is minimized. If the constraints are not convex, then this minimization problem is likely to have infeasible local minimizers regardless of the existence of a feasible point. It follows that the minimization method may terminate at a “phantom” infeasible point even though the constraints have a feasible point (see Section 1.3).

Sequential quadratic programming methods and interior methods are two alternative approaches to handling the inequality constraints in (1.1). Sequential quadratic programming (SQP) methods find an approximate solution of a sequence of quadratic programming (QP) subproblems in which a quadratic model of the objective function is minimized subject to the linearized constraints. Interior methods approximate a continuous path that passes through a solution of (1.1). In the simplest case, the path is parameterized by a positive scalar parameter μ that may be interpreted as a perturbation for the optimality conditions for the problem (1.1). Both interior methods and SQP methods have an inner/outer iteration structure, with the work for an inner iteration being dominated by the cost of solving a large sparse system of symmetric indefinite linear equations. In the case of SQP methods, these equations involve a subset of the variables and constraints; for interior methods, the equations involve all the constraints and variables.

SQP methods provide a relatively reliable “certificate of infeasibility” and they have the potential of being able to capitalize on a good initial starting point. Sophisticated matrix factorization updating techniques are used to exploit the fact that the linear equations change by only a single row and column at each inner iteration. These updating techniques are often customized for the particular QP method being used and have the benefit of providing a uniform treatment of ill-conditioning and singularity.

On the negative side, it is difficult to implement SQP methods so that exact second derivatives can be used efficiently and reliably. Some of these difficulties stem from the theoretical properties of the quadratic programming subproblem, which can be nonconvex when second derivatives are used. Nonconvex quadratic programming is NP-hard—even for the calculation of a local minimizer [43, 72]. The complexity of the QP subproblem has been a major impediment to the formulation of second-derivative SQP methods (although methods based on indefinite QP have been proposed [63, 66]). Over the years, algorithm developers have avoided this difficulty by eschewing second derivatives and by solving a convex QP subproblem defined with a positive semidefinite quasi-Newton approximate Hessian (see, e.g., [83]). There are other difficulties associated with conventional SQP methods that are not specifically related to the use of second derivatives. An SQP algorithm is often tailored to a particular updating technique, e.g., the matrix factors of the Jacobian in the outer iteration can be chosen to match those of the method for the QP subproblem. Any reliance on customized linear algebra software makes it hard to “modernize” a method to reflect new developments in software technology (e.g., in languages that exploit new advances in computer hardware such as multicore processors or GPU-based architectures). Another difficulty is that active-set methods may require a substantial number of QP iterations when the outer iterates are far from the solution. The use of a QP subproblem is motivated by the assumption that the QP objective and constraints provide good “models” of the objective and constraints of the NLP (see Section 2). This should make it unnecessary (and inefficient) to solve the QP to high accuracy during the preliminary iterations. Unfortunately, the simple expedient of limiting the number of inner iterations may have a detrimental effect upon reliability. An approximate QP solution may not predict a sufficient improvement in a merit function (see Section 3.2). Moreover, some of the QP multipliers will have the wrong sign if an active-set method is terminated before a solution is found. This may cause difficulties if the QP multipliers are used to estimate the multipliers for the nonlinear problem. These issues would largely disappear if a primal-dual *interior* method were to be used to solve the QP subproblem. These methods have the benefit of providing a sequence of feasible (i.e., correctly signed) dual iterates. Nevertheless, QP solvers based on conventional interior methods have had limited success within SQP methods because they are difficult to “warm start” from a near-optimal point (see the discussion below). This makes it difficult to capitalize on the property that, as the outer iterates converge, the solution of one QP subproblem is a very good estimate of the solution of the next.

Broadly speaking, the advantages and disadvantages of SQP methods and interior methods complement each other. Interior methods are most efficient when implemented with exact second derivatives. Moreover, they can converge in few inner iterations—even for very large problems. The inner iterates are the iterates of Newton’s method for finding an approximate

solution of the perturbed optimality conditions for a given μ . As the dimension and zero/nonzero structure of the Newton equations remains *fixed*, these Newton equations may be solved efficiently using either iterative or direct methods available in the form of advanced “off-the-shelf” linear algebra software. In particular, any new software for multicore and parallel architectures is immediately applicable. Moreover, the perturbation parameter μ plays an auxiliary role as an implicit regularization parameter of the linear equations. This implicit regularization plays a crucial role in the robustness of interior methods on ill-conditioned and ill-posed problems.

On the negative side, although interior methods are very effective for solving “one-off” problems, they are difficult to adapt to solving a sequence of related NLP problems. This difficulty may be explained in terms of the “path-following” interpretation of interior methods. In the neighborhood of an optimal solution, a step *along* the path $x(\mu)$ of perturbed solutions is well-defined, whereas a step *onto* the path from a neighboring point will be extremely sensitive to perturbations in the problem functions (and hence difficult to compute). Another difficulty with conventional interior methods is that a substantial number of iterations may be needed when the constraints are infeasible.

1.1. Notation. Given vectors a and b with the same dimension, the vector with i th component $a_i b_i$ is denoted by $a \cdot b$. The vectors e and e_j denote, respectively, the column vector of ones and the j th column of the identity matrix I . The dimensions of e , e_i and I are defined by the context. Given vectors x and y of dimension n_x and n_y , the $(n_x + n_y)$ -vector of elements of x augmented by elements of y is denoted by (x, y) . The i th component of a vector labeled with a subscript will be denoted by $(\cdot)_i$, e.g., $(v_N)_i$ is the i th component of the vector v_N . Similarly, the subvector of components with indices in the index set \mathcal{S} is denoted by $(\cdot)_{\mathcal{S}}$, e.g., $(v_N)_{\mathcal{S}}$ is the vector with components $(v_N)_i$ for $i \in \mathcal{S}$. The vector with components $\max\{-x_i, 0\}$ (i.e., the magnitude of the negative part of x) is denoted by $[x]_-$. The vector p -norm and its subordinate matrix norm is denoted by $\|\cdot\|_p$.

1.2. Background. To simplify the notation, the problem format of (1.1) is modified by introducing slack variables and replacing each general constraint of the form $\ell_i \leq \varphi_i(x) \leq u_i$ by the equality constraint $\varphi_i(x) - s_i = 0$ and range constraint $\ell_i \leq s_i \leq u_i$. Without loss of generality, we assume only nonnegativity constraints and use $c(x)$ to denote the vector of combined linear and nonlinear equality constraint functions. (However, we emphasize that the exploitation of the properties of linear constraints is an important issue in the solution of MINLP problems.) The problem to be solved is then

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = 0, \quad x \geq 0, \quad (1.2)$$

where f and the m components of the constraint vector c are assumed to be twice continuously differentiable for all $x \in \mathbb{R}^n$. Any slack variables are included in the definition of x and c .

Let $g(x)$ denote $\nabla f(x)$, the gradient of f evaluated at x . Similarly, let $J(x)$ denote the $m \times n$ constraint Jacobian with rows formed from the constraint gradients $\nabla c_i(x)$. It is assumed that $J(x)$ has rank m for all x (see the discussion of the use of slack variables in Section 1). Throughout the discussion, the component π_i of the m -vector π will denote the dual variable associated with the constraint $c_i(x) = 0$ or its linearization. Similarly, z_j denotes the dual variable associated with the bound $x_j \geq 0$.

A constraint is *active* at x if it is satisfied with equality. For any *feasible* x , i.e., for any x such that $c(x) = 0$ and $x \geq 0$, all m equality constraints $c_i(x) = 0$ are necessarily active. The indices associated with the active nonnegativity constraints comprise the *active set*, denoted by $\mathcal{A}(x)$, i.e., $\mathcal{A}(x) = \{i : x_i = 0\}$. A nonnegativity constraint that is not in the active set is said to be *inactive*. The *inactive set* contains the indices of the inactive constraints, i.e., the so-called “free” variables $\mathcal{I}(x) = \{i : x_i > 0\}$.

Under certain constraint regularity assumptions, an optimal solution of (1.2) must satisfy conditions that may be written in terms of the derivatives of the Lagrangian function $L(x, \pi, z) = f(x) - \pi^T c(x) - z^T x$. The triple (x^*, π^*, z^*) is said to be a first-order KKT point for problem (1.2) if it satisfies the KKT conditions

$$\begin{aligned} c(x^*) &= 0, & x^* &\geq 0, \\ g(x^*) - J(x^*)^T \pi^* - z^* &= 0, & (1.3) \\ x^* \cdot z^* &= 0, & z^* &\geq 0. \end{aligned}$$

The property of strict complementarity holds if the vectors x^* and z^* satisfy $x^* \cdot z^* = 0$ with $x^* + z^* > 0$. In keeping with linear programming terminology, we refer to the dual variables π and z as the *π -values* and *reduced costs*, respectively. The vector-triple (x, π, z) is said to constitute a *primal-dual estimate* of the quantities (x^*, π^*, z^*) satisfying (1.3).

The purpose of the constraint regularity assumption is to guarantee that a linearization of the constraints describes the nonlinear constraints with sufficient accuracy that the KKT conditions of (1.3) are necessary for local optimality. One such regularity assumption is the *Mangasarian-Fromovitz constraint qualification* [133, 143], which requires that $J(x^*)$ has rank m , and that there exists a vector p such that $J(x^*)p = 0$ with $p_i > 0$ for all $i \in \mathcal{A}(x^*)$. Another common, but slightly more restrictive, assumption is the *linear independence constraint qualification*, which requires that the matrix of free columns of $J(x^*)$ has full row rank.

Let $H(x, \pi)$ denote the Hessian of $L(x, \pi, z)$ with respect to x , i.e.,

$$H(x, \pi) = \nabla_{xx}^2 L(x, \pi, z) = \nabla^2 f(x) - \sum_{i=1}^m \pi_i \nabla^2 c_i(x).$$

Under the linear independence constraint qualification, the second-order necessary optimality conditions require that the first-order conditions (1.3) hold with the additional condition that $p^T H(x^*, \pi^*)p \geq 0$ for all p such that $J(x^*)p = 0$, and $p_i = 0$ for every $i \in \mathcal{A}(x^*)$. See, e.g., Nocedal and Wright [143, Chapter 12] for more discussion of constraint assumptions and optimality conditions.

For a feasible point x , we will denote by $\widehat{J}(x)$ the matrix comprising columns of $J(x)$ corresponding to indices in $\mathcal{I}(x)$. A point x at which $(g(x))_{\mathcal{I}} \in \text{range}(\widehat{J}(x)^T)$ and the linear independence constraint qualification does not hold is said to be *degenerate*. For example, if x is a degenerate vertex, then more than $n - m$ bounds must be active and $\widehat{J}(x)$ has more rows than columns. The Mangasarian-Fromovitz constraint qualification may or may not hold at a degenerate point. Practical NLP problems with degenerate points are very common and it is crucial that an algorithm be able to handle $\widehat{J}(x)$ with dependent rows. Throughout our discussion of the effects of degeneracy in SQP methods, it will be assumed that the Mangasarian-Fromovitz regularity assumption holds.

1.3. Infeasible problems. In the normal situation, when solving a “one-off” nonlinear program of the form (1.2), one may expect that the problem is feasible—i.e., that there exist points that satisfy the constraints. This is because an infeasible problem is generally the result of a unintended formulation or coding error. However, there are situations when the detection of infeasibility is of particular interest. An example is mixed integer nonlinear programming, where the occurrence of infeasibility is likely as part of a branch and bound fathoming criteria. In this situation, the rapid and reliable detection of infeasibility is a crucial requirement of an algorithm. One way of handling this situation is to define a related *regularized* problem that always has feasible points. This is done by formulating an alternative problem that is always well posed, yet has (x^*, π^*, z^*) as a solution when (x^*, π^*, z^*) exists.

As the question of the existence of a solution revolves around whether or not the constraints admit a feasible point, we can always *relax* the constraints sufficiently to allow the constraints to be feasible. It is then just a question of solving the relaxed problem while simultaneously reducing the amount of relaxation. This process can be automated by introducing *elastic variables* v and w in (1.2), and formulating the *elastic problem*

$$\begin{aligned} & \underset{x \in \mathbb{R}^n; u, v \in \mathbb{R}^m}{\text{minimize}} && f(x) + \rho e^T u + \rho e^T v \\ & \text{subject to} && c(x) - u + v = 0, \quad x \geq 0, \quad u \geq 0, \quad v \geq 0, \end{aligned} \tag{1.4}$$

where ρ is a “penalty” on the elasticity, often referred to as the *elastic weight*, and e is the vector of ones. The smooth elastic problem is equivalent to the nonsmooth bound-constrained problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) + \rho \sum_{i=1}^m |c_i(x)| \quad \text{subject to } x \geq 0, \quad (1.5)$$

i.e., the elastic problem implicitly enforces a penalty on the sum of the infeasibilities of the constraints $c(x) = 0$. If the original problem is infeasible, then, for large values of ρ , there is a minimizer of the elastic problem that is an $O(1/\rho)$ approximation to a minimizer of the sum of the constraint infeasibilities. This minimizer can be useful in identifying which of the constraints are causing the infeasibility (see Chinneck [34, 35]).

The elastic problem is called an *exact regularization* of (1.2) because if ρ is sufficiently large and (x^*, π^*, z^*) is optimal for (1.2), then it is also optimal for the elastic problem (1.4) with $u = v = 0$. See Fletcher [64, Section 12.3] for a discussion of these issues. The first-order necessary conditions for $(x^*, u^*, v^*, \pi^*, z^*)$ to be an optimal solution for the elastic problem (1.4) are

$$c(x^*) - u^* + v^* = 0, \quad u^* \geq 0, \quad v^* \geq 0, \quad (1.6a)$$

$$g(x^*) - J(x^*)^T \pi^* - z^* = 0, \quad (1.6b)$$

$$x^* \cdot z^* = 0, \quad z^* \geq 0, \quad x^* \geq 0 \quad (1.6c)$$

$$u^* \cdot (\rho e + \pi^*) = 0, \quad v^* \cdot (\rho e - \pi^*) = 0, \quad -\rho e \leq \pi^* \leq \rho e. \quad (1.6d)$$

To see that the elastic problem (1.4) defines an exact regularization, note that if $\|\pi^*\|_\infty < \rho$, then a solution (x^*, π^*, z^*) of (1.3) is also a solution of (1.6) with $u^* = v^* = 0$. Conditions (1.6) are always necessary for a point (x^*, u^*, v^*) to be an optimal solution for (1.4) because the Mangasarian-Fromovitz constraint qualification is always satisfied.

There are two caveats associated with solving the regularized problem. First, if a solution of the original problem exists, it is generally only a *local* solution of the elastic problem. The elastic problem may be unbounded below, or may have local solutions that are not solutions of the original problem. For example, consider the one-dimensional problem

$$\underset{x \in \mathbb{R}}{\text{minimize}} \quad x + 1 \quad \text{subject to } \frac{1}{3}x^3 - \frac{3}{2}x^2 + 2x = 0, \quad x \geq 0, \quad (1.7)$$

which has a unique solution $(x^*, \pi^*) = (0, \frac{1}{2})$. For all $\rho > \frac{1}{2}$, the penalty function (1.5) has a local minimizer $\bar{x} = 2 - O(1/\rho)$ such that $c(\bar{x}) \neq 0$. This example shows that regularization can introduce “phantom” solutions that do not appear in the original problem.

The second caveat is that, in general, the precise definition of the elastic problem is not known in advance because an appropriate value of the parameter ρ depends on the optimal multipliers π^* . This implies that, in practice, any estimate of ρ may need to be increased if the minimization appears to be converging to a regularized solution with $u^* + v^* \neq 0$. If the

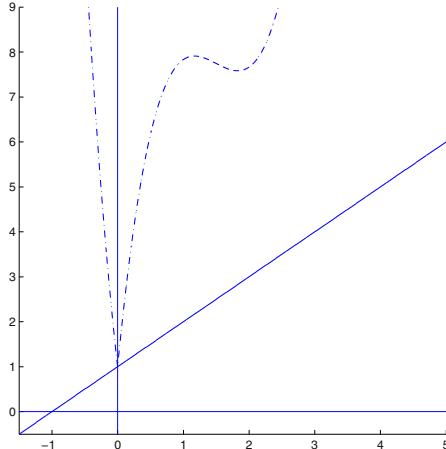


FIG. 1. This figure depicts the objective function and penalty function (1.5) for the one-dimensional problem (1.7). The constrained problem has a unique solution $(x^*, \pi^*) = (0, \frac{1}{2})$. However, for all $\rho > \frac{1}{2}$, the penalty function has a local minimizer $\bar{x} = 2 - O(1/\rho)$ with $c(\bar{x}) \neq 0$.

original problem is infeasible, then $u^* + v^*$ is nonzero for all solutions and ρ must go to infinity if the elastic problem is to provide an estimate of the minimum sum of infeasibilities. Gill, Murray, and Saunders [83] apply an SQP method to a sequence of regularized problems in which ρ is increased geometrically. The sequence is terminated when a solution is found with $u^* + v^* = 0$, or ρ reaches a preassigned upper limit. However, in the context of MINLP, where constraint infeasibility is typical, it is crucial that infeasibility be detected rapidly, which implies that the value of ρ may need to be increased before an accurate solution of the regularized problem is found. (For further information on choosing the value of ρ in this context, see, e.g., Exler and Schittkowski [57], and Byrd, Nocedal, and Waltz [28].)

The objective function in the elastic problem (1.5) is the ℓ_1 penalty function

$$P_1(x; \rho) = f(x) + \rho \|c(x)\|_1 = f(x) + \rho \sum_{i=1}^m |c_i(x)|. \quad (1.8)$$

Regularization using an ℓ_1 penalty function is (by far) the most common form of constraint regularization for problems with inequality constraints. However, other exact regularizations can be defined based on using alternative norms to measure the constraint violations. If the ℓ_∞ penalty function

$$P_\infty(x; \rho) = f(x) + \rho \|c(x)\|_\infty = f(x) + \rho \max_{1 \leq i \leq m} |c_i(x)| \quad (1.9)$$

is minimized subject to the constraints $x \geq 0$, an equivalent smooth constrained form of the regularized problem is

$$\underset{x \in \mathbb{R}^n; \theta \in \mathbb{R}}{\text{minimize}} \quad f(x) + \rho\theta \quad \text{subject to} \quad -\theta e \leq c(x) \leq \theta e, \quad x \geq 0, \quad \theta \geq 0, \quad (1.10)$$

where θ is a temporary nonnegative auxiliary variable. This regularization is exact if $\rho > \|\pi^*\|_1$. The ℓ_2 penalty function $f(x) + \rho\|c(x)\|_2$ also defines an exact regularization, although the use of the two-norm in this form is less common because there is no equivalent smooth constrained form of the problem. (For more on the properties of exact regularization for convex optimization, see Friedlander and Tseng [76].)

The ℓ_2 penalty function is one exception to the rule that a constraint regularization for (1.2) can be written as either a perturbed nonlinearly constrained problem or an equivalent bound-constrained problem, where both formulations depend on the optimal multipliers π^* . However, for some forms of regularization, the dependence on π^* can be explicit (and hence harder to apply). Consider the bound-constrained problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) - c(x)^T \pi_E + \frac{1}{2}\rho\|c(x)\|_2^2 \quad \text{subject to} \quad x \geq 0, \quad (1.11)$$

where π_E is an m -vector, and ρ is a nonnegative scalar penalty parameter. Problem (1.11) is used in a number of methods for general nonlinear programming problems based on sequential bound-constrained minimization, see, e.g., [39, 40, 41, 73, 75]. The objective function is the well-known Hestenes-Powell augmented Lagrangian, which was first proposed for sequential unconstrained optimization (see, e.g., Hestenes [122], Powell [151], Rockafellar [158], Tapia [166], and Bertsekas [6]). The regularization is exact for $\pi_E = \pi^*$ and all $\rho > \bar{\rho}$, where $\bar{\rho}$ depends on the spectral radius of the Hessian of the Lagrangian (and hence, implicitly, on the magnitude of $\|\pi^*\|$). Clearly, this function has a more explicit dependence on π^* . If x_* is a solution of (1.11) for $\pi_E \approx \pi^*$, then x_* satisfies the perturbed nonlinearly constrained problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = \mu(\pi_E - \pi_*), \quad x \geq 0, \quad (1.12)$$

where $\pi_* = \pi_E - \rho c(x_*)$ and μ is the inverse penalty parameter $1/\rho$.

In later sections we consider the close connection between problem regularization and the formulation of SQP methods. In particular, we show that many SQP formulations can be considered in terms of a “plain” SQP method being applied to a related regularized nonlinear program.

Regularization is a very broad idea that can take many forms. For example, a problem formulation may be regularized to ensure that a solution exists or, if there are many solutions, to ensure that a particular favorable solution is found (such as a least-norm solution). Other forms of regularization are specifically designed to make sure that an algorithm

may be applied with minimal risk of numerical breakdown. For example, adding slack variables to all constraints, including equalities, guarantees that the Jacobian has full row rank. Such regularization schemes have a beneficial effect on whatever method is used. Some forms of regularization are associated with a specific technique (e.g., trust-region methods impose an implicit regularization on a given subproblem—see Section 3.1.2).

However, although regularization is useful (and sometimes vital) there is usually some price to be paid for its use. In many cases, regularization leads to additional computational overhead or algorithmic complexity. In some cases, regularization will give an approximate rather than exact solution of the problem. More seriously, some forms of regularization lead to the possibility of “phantom” solutions that are not solutions of the original problem.

2. Local properties of SQP methods. In many introductory texts, “the” SQP method is defined as one in which the quadratic programming subproblem involves the minimization of a quadratic model of the objective function subject to a linearization of the constraints. This description, which broadly defines the original SQP method of Wilson [172] for convex programming, is somewhat over-simplistic for modern SQP methods. Nevertheless, we start by defining a “vanilla” or “plain” SQP method in these terms.

The basic structure of an SQP method involves *inner* and *outer* iterations. Associated with the k th outer iteration is an approximate solution x_k , together with dual variables π_k and z_k for the nonlinear constraints and bounds. Given (x_k, π_k, z_k) , new primal-dual estimates are found by solving the quadratic programming subproblem

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} && f(x_k) + g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k, \pi_k)(x - x_k) \\ & \text{subject to} && c(x_k) + J(x_k)(x - x_k) = 0, \quad x \geq 0. \end{aligned} \tag{2.1}$$

In our plain SQP method, this subproblem is solved by iteration using a quadratic programming method. New estimates π_{k+1} and z_{k+1} of the Lagrange multipliers are the optimal multipliers for the subproblem (2.1). The iterations of the QP method constitute the SQP inner iterations.

The form of the plain QP subproblem (2.1) is motivated by a certain *fixed-point property* that requires the SQP method to terminate in only one (outer) iteration when started at an optimal solution. In particular, the plain QP subproblem is defined in such a way that if $(x_k, \pi_k, z_k) = (x^*, \pi^*, z^*)$, then the NLP primal-dual solution (x^*, π^*, z^*) satisfies the QP optimality conditions for (2.1) and thereby constitutes a solution of the subproblem (see Section 2.2 below for a statement of the QP optimality conditions). Under certain assumptions on the problem derivatives, this fixed-point property implies that $(x_k, \pi_k, z_k) \rightarrow (x^*, \pi^*, z^*)$ when the initial point (x_0, π_0, z_0) is sufficiently close to (x^*, π^*, z^*) . These assumptions are discussed further below.

Given our earlier statement that SQP methods “minimize a quadratic model of the objective function”, readers unfamiliar with SQP methods might wonder why the quadratic term of the quadratic objective of (2.1) involves the Hessian of the Lagrangian function and not the Hessian of the objective function. However, at $(x_k, \pi_k, z_k) = (x^*, \pi^*, z^*)$, the objective of the subproblem defines the second-order local variation of f on the constraint surface $c(x) = 0$. Suppose that $x(\alpha)$ is a twice-differentiable feasible path starting at x_k , parameterized by a nonnegative scalar α ; i.e., $x(0) = x_k$ and $c(x(\alpha)) = 0$. An inspection of the derivatives $f'(x(\alpha))$ and $f''(x(\alpha))$ at $\alpha = 0$ indicates that the function

$$\hat{f}_k(x) = f(x_k) + g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k, \pi_k)(x - x_k) \quad (2.2)$$

defines a second-order approximation of f for all x lying on $x(\alpha)$, i.e., $\hat{f}_k(x)$ may be regarded as a *local quadratic model of f that incorporates the curvature of the constraints $c(x) = 0$* .

This constrained variation of the objective is equivalent to the *unconstrained* variation of a function known as the *modified Lagrangian*, which is given by

$$L(x; x_k, \pi_k) = f(x) - \pi_k^T(c(x) - \hat{c}_k(x)), \quad (2.3)$$

where $\hat{c}_k(x)$ denotes the vector of linearized constraint functions $\hat{c}_k(x) = c(x_k) + J(x_k)(x - x_k)$, and $c(x) - \hat{c}_k(x)$ is known as the *departure from linearity* (see Robinson [156] and Van der Hoek [169]). The first and second derivatives of the modified Lagrangian are given by

$$\begin{aligned} \nabla L(x; x_k, \pi_k) &= g(x) - (J(x) - J(x_k))^T \pi_k, \\ \nabla^2 L(x; x_k, \pi_k) &= \nabla^2 f(x) - \sum_{i=1}^m (\pi_k)_i \nabla^2 c_i(x). \end{aligned}$$

The Hessian of the modified Lagrangian is independent of x_k and coincides with the Hessian (with respect to x) of the conventional Lagrangian. Also, $L(x; x_k, \pi_k)|_{x=x_k} = f(x_k)$, and $\nabla L(x; x_k, \pi_k)|_{x=x_k} = g(x_k)$, which implies that $\hat{f}_k(x)$ defines a local quadratic model of $L(x; x_k, \pi_k)$ at $x = x_k$.

Throughout the remaining discussion, g_k , c_k , J_k and H_k denote $g(x)$, $c(x)$, $J(x)$ and $H(x, \pi)$ evaluated at x_k and π_k . With this notation, the quadratic objective is $\hat{f}_k(x) = f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k(x - x_k)$, with gradient $\hat{g}_k(x) = g_k + H_k(x - x_k)$. A “hat” will be used to denote quantities associated with the QP subproblem.

2.1. Equality constraints. We motivate some of the later discussion by reviewing the connection between SQP methods and Newton’s method for solving a system of nonlinear equations. We begin by omitting the

nonnegativity constraints and considering the equality constrained problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = 0. \quad (2.4)$$

In the case of unconstrained optimization, a standard approach to the formulation of algorithms is to use the first-order optimality conditions to define a system of nonlinear equations $\nabla f(x) = 0$ whose solution is a first-order optimal point x^* . In the constrained case, the relevant nonlinear equations involve the gradient of the Lagrangian function $L(x, \pi)$, which incorporates the first-order feasibility and optimality conditions satisfied by x^* and π^* . If the rows of the constraint Jacobian J at x^* are linearly independent, a primal-dual solution represented by the $n+m$ vector (x^*, π^*) must satisfy the $n+m$ nonlinear equations $F(x, \pi) = 0$, where

$$F(x, \pi) \equiv \nabla L(x, \pi) = \begin{pmatrix} g(x) - J(x)^T \pi \\ -c(x) \end{pmatrix}. \quad (2.5)$$

These equations may be solved efficiently using Newton's method.

2.1.1. Newton's method and SQP. Consider one iteration of Newton's method, starting at estimates x_k and π_k of the primal and dual variables. If v_k denotes the iterate defined by $(n+m)$ -vector (x_k, π_k) , then the next iterate v_{k+1} is given by

$$v_{k+1} = v_k + \Delta v_k, \quad \text{where } F'(v_k) \Delta v_k = -F(v_k).$$

Differentiating (2.5) with respect to x and π gives $F'(v) \equiv F'(x, \pi)$ as

$$F'(x, \pi) = \begin{pmatrix} H(x, \pi) & -J(x)^T \\ -J(x) & 0 \end{pmatrix},$$

which implies that the Newton equations may be written as

$$\begin{pmatrix} H_k & -J_k^T \\ -J_k & 0 \end{pmatrix} \begin{pmatrix} p_k \\ q_k \end{pmatrix} = - \begin{pmatrix} g_k - J_k^T \pi_k \\ -c_k \end{pmatrix},$$

where p_k and q_k denote the Newton steps for the primal and dual variables. If the second block of equations is scaled by -1 we obtain the system

$$\begin{pmatrix} H_k & -J_k^T \\ J_k & 0 \end{pmatrix} \begin{pmatrix} p_k \\ q_k \end{pmatrix} = - \begin{pmatrix} g_k - J_k^T \pi_k \\ c_k \end{pmatrix}, \quad (2.6)$$

which is an example of a *saddle-point system*. Finally, if the second block of variables is scaled by -1 we obtain an equivalent symmetric system

$$\begin{pmatrix} H_k & J_k^T \\ J_k & 0 \end{pmatrix} \begin{pmatrix} p_k \\ -q_k \end{pmatrix} = - \begin{pmatrix} g_k - J_k^T \pi_k \\ c_k \end{pmatrix}, \quad (2.7)$$

which is often referred to as the *KKT system*.

It may not be clear immediately how this method is related to an SQP method. The crucial link follows from the observation that the KKT equations (2.7) represent the first-order optimality conditions for the primal and dual solution (p_k, q_k) of the quadratic program

$$\begin{aligned} & \underset{p \in \mathbb{R}^n}{\text{minimize}} \quad (g_k - J_k^T \pi_k)^T p + \frac{1}{2} p^T H_k p \\ & \text{subject to} \quad c_k + J_k p = 0, \end{aligned}$$

which, under certain conditions on the curvature of the Lagrangian discussed below, defines the step from x_k to the point that minimizes the local quadratic model of the objective function subject to the linearized constraints. It is now a simple matter to include the constant objective term f_k (which does not affect the optimal solution) and write the dual variables in terms of $\pi_{k+1} = \pi_k + q_k$ instead of q_k . The equations analogous to (2.7) are then

$$\begin{pmatrix} H_k & J_k^T \\ J_k & 0 \end{pmatrix} \begin{pmatrix} p_k \\ -\pi_{k+1} \end{pmatrix} = - \begin{pmatrix} g_k \\ c_k \end{pmatrix}, \quad (2.8)$$

which are the first-order optimality conditions for the quadratic program

$$\underset{p \in \mathbb{R}^n}{\text{minimize}} \quad f_k + g_k^T p + \frac{1}{2} p^T H_k p \quad \text{subject to} \quad c_k + J_k p = 0.$$

When written in terms of the x variables, this quadratic program is

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k(x - x_k) \\ & \text{subject to} \quad c_k + J_k(x - x_k) = 0. \end{aligned} \quad (2.9)$$

2.1.2. Local convergence. A standard analysis of Newton's method (see, e.g., Moré and Sorensen [139, Theorem 2.8]) shows that if the KKT matrix is nonsingular at a solution (x^*, π^*) , and (x_0, π_0) lies in a sufficiently small neighborhood of (x^*, π^*) in which f and c are twice-continuously differentiable, then the SQP iterates (x_k, π_k) will converge to (x^*, π^*) at a Q-superlinear rate. If, in addition, $H(x, \pi)$ is locally Lipschitz continuous, then the SQP iterates (x_k, π_k) are Q-quadratically convergent. As x is only a subvector of v , with $v = (x, \pi)$, the convergence rate of x_k does not follow immediately. However, as $\|x_k - x^*\| \leq \|v_k - v^*\|$, a Q-quadratic rate of convergence of (x_k, π_k) implies an R-quadratic rate of convergence of x_k . For more on the rate of convergence of $\{x_k\}$ relative to $\{x_k, \pi_k\}$, see Ortega and Rheinboldt [146, Chapter 9].

Conditions for the nonsingularity of the KKT matrix may be determined by transforming the KKT system into an equivalent system that reveals the rank. If Q_k is an $n \times n$ nonsingular matrix, then (2.8) is equivalent to the system

$$\begin{pmatrix} Q_k^T H_k Q_k & (J_k Q_k)^T \\ J_k Q_k & 0 \end{pmatrix} \begin{pmatrix} p_Q \\ -\pi_{k+1} \end{pmatrix} = - \begin{pmatrix} Q_k^T g_k \\ c_k \end{pmatrix}, \quad \text{with } p_k = Q_k p_Q. \quad (2.10)$$

Let Q_k be defined so that $J_k Q_k = (0 \ U_k)$, where U_k is $m \times m$. The assumption that J_k has rank m implies that U_k is nonsingular. If the n columns of Q_k are partitioned into blocks Z_k and Y_k of dimension $n \times (n-m)$ and $n \times m$, then

$$J_k Q_k = J_k \begin{pmatrix} Z_k & Y_k \end{pmatrix} = \begin{pmatrix} 0 & U_k \end{pmatrix}, \quad (2.11)$$

which shows that $J_k Z_k = 0$ and $J_k Y_k = U_k$. Since Z_k and Y_k are sections of the nonsingular matrix Q_k , they must have independent columns, and, in particular, the columns of Z_k must form a basis for the null-space of J_k . If $Q_k^T H_k Q_k$ and $J_k Q_k$ are partitioned to conform to the $Z-Y$ partition of Q_k , we obtain the block lower-triangular system

$$\begin{pmatrix} U_k & 0 & 0 \\ Z_k^T H_k Y_k & Z_k^T H_k Z_k & 0 \\ Y_k^T H_k Y_k & Y_k^T H_k Z_k & U_k^T \end{pmatrix} \begin{pmatrix} p_Y \\ p_Z \\ -\pi_{k+1} \end{pmatrix} = - \begin{pmatrix} c_k \\ Z_k^T g_k \\ Y_k^T g_k \end{pmatrix}, \quad (2.12)$$

where the $(n-m)$ -vector p_Z and m -vector p_Y are the parts of p_Q that conform to the columns of Z_k and Y_k . It follows immediately from (2.12) that the Jacobian $F'(x_k, \pi_k)$ is nonsingular if J_k has independent rows and $Z_k^T H_k Z_k$ is nonsingular. In what follows, we use standard terminology and refer to the vector $Z_k^T g_k$ as the *reduced gradient* and the matrix $Z_k^T H_k Z_k$ as the *reduced Hessian*. If $J(x^*)$ has rank m and the columns of the matrix Z^* form a basis for the null-space of $J(x^*)$, then the conditions: (i) $\nabla L(x^*, \pi^*) = 0$; and (ii) $Z^{*T} H(x^*, \pi^*) Z^*$ positive definite, are sufficient for x^* to be an isolated minimizer of the equality constraint problem (2.4).

2.1.3. Properties of the Newton step. The equations (2.12) have a geometrical interpretation that provides some insight into the properties of the Newton direction. From (2.10), the vectors p_Z and p_Y must satisfy

$$p_k = Q_k p_Q = \begin{pmatrix} Z_k & Y_k \end{pmatrix} \begin{pmatrix} p_Z \\ p_Y \end{pmatrix} = Z_k p_Z + Y_k p_Y.$$

Using block substitution on the system (2.12) we obtain the following equations for p_k and π_{k+1} :

$$\begin{aligned} U_k p_Y &= -c_k, & p_N &= Y_k p_Y, \\ Z_k^T H_k Z_k p_Z &= -Z_k^T(g_k + H_k p_N), & p_T &= Z_k p_Z, \\ p_k = p_N + p_T, & \quad U_k^T \pi_{k+1} = Y_k^T(g_k + H_k p_k). \end{aligned} \quad (2.13)$$

These equations involve the auxiliary vectors p_N and p_T such that $p_k = p_N + p_T$ and $J_k p_T = 0$. We call p_N and p_T the *normal* and *tangential* steps associated with p_k . Equations (2.13) may be simplified further by introducing the intermediate vector x_F such that $x_F = x_k + p_N$. The definition of the gradient of \hat{f}_k implies that $g_k + H_k p_N = \nabla f_k(x_k + p_N) = \hat{g}_k(x_F)$, which allows us to rewrite (2.13) in the form

$$\begin{aligned}
U_k p_Y &= -c_k, & p_N &= Y_k p_Y, \\
x_F = x_k + p_N, \quad Z_k^T H_k Z_k p_Z &= -Z_k^T \hat{g}_k(x_F), & p_T &= Z_k p_Z, \\
p_k = p_N + p_T, \quad x_{k+1} &= x_F + p_T, \\
U_k^T \pi_{k+1} &= Y_k^T \hat{g}_k(x_{k+1}).
\end{aligned} \tag{2.14}$$

The definition of x_F implies that

$$\hat{c}_k(x_F) = c_k + J_k(x_F - x_k) = c_k + J_k p_N = c_k + J_k Y_k p_Y = c_k + U_k p_Y = 0,$$

which implies that the normal component p_N satisfies $J_k p_N = -c_k$ and constitutes the Newton step from x_k to the point x_F satisfying the linearized constraints $c_k + J_k(x - x_k) = 0$. On the other hand, the tangential step p_T satisfies $p_T = Z_k p_Z$, where $Z_k^T H_k Z_k p_Z = -Z_k^T \hat{g}_k(x_F)$. If the reduced Hessian $Z_k^T H_k Z_k$ is positive definite, which will be the case if x_k is sufficiently close to a locally unique (i.e., isolated) minimizer of (2.4), then p_T defines the Newton step from x_F to the *minimizer* of the quadratic model $\hat{f}_k(x)$ in the subspace orthogonal to the constraint normals (i.e., on the surface of the linearized constraint $\hat{c}_k(x) = 0$). It follows that the Newton direction is the sum of two steps: a normal step to the linearized constraint and the tangential step on the constraint surface that minimizes the quadratic model. This property reflects the two (usually conflicting) underlying processes present in all algorithms for optimization—the minimization of the objective and the satisfaction of the constraints.

In the discussion above, the normal step p_N is interpreted as a Newton direction for the equations $\hat{c}_k(x) = 0$ at $x = x_k$. However, in some situations, p_N may also be interpreted as the solution of a minimization problem. The Newton direction p_k is unique, but the decomposition $p_k = p_T + p_N$ depends on the choice of the matrix Q_k associated with the Jacobian factorization (2.11). If Q_k is orthogonal, i.e., if $Q_k^T Q_k = I$, then $Z_k^T Y_k = 0$ and the columns of Y_k form a basis for the range space of J_k^T . In this case, p_N and p_T define the unique range-space and null-space decomposition of p_k , and p_N is the unique solution with least two-norm of the least-squares problem

$$\min_p \|\hat{c}_k(x_k) + J_k p\|_2, \quad \text{or, equivalently,} \quad \min_p \|c_k + J_k p\|_2.$$

This interpretation is useful in the formulation of variants of Newton's method that do not require (x_k, π_k) to lie in a small neighborhood of (x^*, π^*) . In particular, it suggests a way of computing the normal step when the equations $J_k p = -c_k$ are not compatible.

For consistency with the inequality constrained case below, the primal-dual solution of the k th QP subproblem is denoted by $(\hat{x}_k, \hat{\pi}_k)$. With this notation, the first-order optimality conditions for the QP subproblem (2.9) are given by

$$\begin{aligned} J_k(\hat{x}_k - x_k) + c_k &= 0, \\ g_k + H_k(\hat{x}_k - x_k) - J_k^T \hat{\pi}_k &= 0. \end{aligned} \quad (2.15)$$

Similarly, the Newton iterates are given by $x_{k+1} = \hat{x}_k = x_k + p_k$ and $\pi_{k+1} = \hat{\pi}_k = \pi_k + q_k$.

2.1.4. Calculation of the Newton step. There are two broad approaches for solving the Newton equations (either in saddle-point form (2.6) or symmetric form (2.7)). The first involves solving the full $n+m$ KKT equations, the second decomposes the KKT equations into the three systems associated with the block lower-triangular equations (2.12).

In the full-matrix approach, the matrix K may be represented by its *symmetric indefinite factorization* (see, e.g., Bunch and Parlett [20], and Bunch and Kaufman [18]):

$$PKP^T = LDL^T, \quad (2.16)$$

where P is a permutation matrix, L is lower triangular and D is block diagonal, with 1×1 or 2×2 blocks. (The latter are required to retain numerical stability.) Some prominent software packages include MA27 (Duff and Reid [55]), MA57 (Duff [54]), MUMPS (Amestoy et al. [1]), PARDISO (Schenk and Gärtner [159]), and SPOOLES (Ashcraft and Grimes [4]).

The decomposition approach is based on using an explicit or implicit representation of the null-space basis matrix Z_k . When J_k is dense, Z_k is usually computed directly from a *QR* factorization of J_k (see, e.g., Coleman and Sorensen [38], and Gill et al. [85]). When J_k is sparse, however, known techniques for obtaining an orthogonal *and sparse* Z may be expensive in time and storage, although some effective algorithms have been proposed (see, e.g., Coleman and Pothen [37]; Gilbert and Heath [78]).

The representation of Z_k most commonly used in sparse problems is called the *variable-reduction* form of Z_k , and is obtained as follows. The columns of J_k are partitioned so as to identify explicitly an $m \times m$ nonsingular matrix B (the *basis matrix*). Assuming that B is at the “left” of J_k , we have

$$J_k = \begin{pmatrix} B & S \end{pmatrix}.$$

(In practice, the columns of B may occur anywhere.) When J_k has this form, a basis for the null space of J_k is given by the columns of the (non-orthogonal) matrix Q_k defined as

$$Q_k = \begin{pmatrix} -B^{-1}S & I_m \\ I_{n-m} & \end{pmatrix}, \text{ with } Z_k = \begin{pmatrix} -B^{-1}S \\ I_{n-m} \end{pmatrix} \text{ and } Y_k = \begin{pmatrix} I_m \\ 0 \end{pmatrix}.$$

This definition of Q_k means that matrix-vector products $Z_k^T v$ or $Z_k v$ can be computed using a factorization of B (typically, a sparse *LU* factorization;

see Gill, Murray, Saunders and Wright [90]), and Z_k need not be stored explicitly.

For large sparse problems, the reduced Hessian $Z_k^T H_k Z_k$ associated with the solution of (2.14) will generally be much more dense than H_k and B . However, in many cases, $n - m$ is small enough to allow the storage of a dense Cholesky factor of $Z_k^T H_k Z_k$.

2.2. Inequality constraints. Given an approximate primal-dual solution (x_k, π_k) with $x_k \geq 0$, an outer iteration of a typical SQP method involves solving the QP subproblem (2.1), repeated here for convenience:

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k(x - x_k) \\ & \text{subject to} \quad J_k(x - x_k) = -c_k, \quad x \geq 0. \end{aligned} \quad (2.17)$$

Assume for the moment that this subproblem is feasible, with primal-dual solution $(\hat{x}_k, \hat{\pi}_k, \hat{z}_k)$. The next plain SQP iterate is $x_{k+1} = \hat{x}_k$, $\pi_{k+1} = \hat{\pi}_k$ and $z_{k+1} = \hat{z}_k$. The QP first-order optimality conditions are

$$\begin{aligned} J_k(\hat{x}_k - x_k) + c_k &= 0, \quad \hat{x}_k \geq 0; \\ g_k + H_k(\hat{x}_k - x_k) - J_k^T \hat{\pi}_k - \hat{z}_k &= 0, \\ \hat{x}_k \cdot \hat{z}_k &= 0, \quad \hat{z}_k \geq 0. \end{aligned} \quad (2.18)$$

Let $p_k = \hat{x}_k - x_k$ and let \bar{p}_k denote the vector of free components of p_k , i.e., the components with indices in $\mathcal{I}(\hat{x}_k)$. Similarly, let \bar{z}_k denote the free components of \hat{z}_k . The complementarity conditions imply that $\bar{z}_k = 0$ and we may combine the first two sets of equalities in (2.18) to give

$$\begin{pmatrix} \bar{H}_k & \bar{J}_k^T \\ \bar{J}_k & 0 \end{pmatrix} \begin{pmatrix} \bar{p}_k \\ -\hat{\pi}_k \end{pmatrix} = - \begin{pmatrix} (g_k + H_k \eta_k)_\mathcal{I} \\ c_k + J_k \eta_k \end{pmatrix}, \quad (2.19)$$

where \bar{J}_k is the matrix of free columns of J_k , and η_k is the vector

$$(\eta_k)_i = \begin{cases} (\hat{x}_k - x_k)_i & \text{if } i \in \mathcal{A}(\hat{x}_k); \\ 0 & \text{if } i \in \mathcal{I}(\hat{x}_k). \end{cases}$$

If the active sets at \hat{x}_k and x_k are the same, i.e., $\mathcal{A}(\hat{x}_k) = \mathcal{A}(x_k)$, then $\eta_k = 0$. If \hat{x}_k lies in a sufficiently small neighborhood of a *nondegenerate* solution x^* , then $\mathcal{A}(\hat{x}_k) = \mathcal{A}(x^*)$ and hence \bar{J}_k has full row rank (see Robinson [157]). In this case we say that the QP *identifies the correct active set* at x^* . If, in addition, (x^*, π^*) satisfies the second-order sufficient conditions for optimality, then KKT system (2.19) is nonsingular and the plain SQP method is equivalent to Newton's method applied to the equality-constraint subproblem defined by fixing the variables in the active set at their bounds.

However, at a *degenerate* QP solution, the rows of \bar{J}_k are linearly dependent and the KKT equations (2.19) are compatible but singular.

Broadly speaking, there are two approaches to dealing with the degenerate case, where each approach is linked to the method used to solve the QP subproblem. The first approach employs a QP method that not only finds the QP solution \hat{x}_k , but also identifies a “basic set” of variables that define a matrix \tilde{J}_k with *linearly independent* rows. The second approach solves a *regularized* or *perturbed* QP subproblem that provides a perturbed version of the KKT system (2.19) that is nonsingular for any \bar{J}_k .

Identifying independent constraints. The first approach is based on using a QP algorithm that provides a primal-dual QP solution that satisfies a *nonsingular* KKT system analogous to (2.19). A class of quadratic programming methods with this property are primal-feasible active-set methods, which form the basis of the software packages NPSOL and SNOPT. Primal-feasible QP methods have two phases: in phase 1, a feasible point is found by minimizing the sum of infeasibilities; in phase 2, the quadratic objective function is minimized while feasibility is maintained. In each iteration, the variables are labeled as being “basic” or “nonbasic”, where the nonbasic variables are temporarily fixed at their current value. The indices of the basic and nonbasic variables are denoted by \mathcal{B} and \mathcal{N} respectively. A defining property of the \mathcal{B} - \mathcal{N} partition is that the rows of the Jacobian appearing in the KKT matrix are always linearly independent. Once an initial basic set is identified, all subsequent KKT equations have a constraint block with independent rows. (For more details of primal-feasible active-set methods, see Section A.1 of the Appendix.)

Let $p_k = \hat{x}_k - x_k$, where $(\hat{x}_k, \hat{\pi}_k)$ is the QP solution found by a primal-feasible active-set method. Let \tilde{p}_k denote the vector of components of p_k in the final basic set \mathcal{B} , with \tilde{J}_k the corresponding columns of J_k . The vector $(\tilde{p}_k, \hat{\pi}_k)$ satisfies the *nonsingular* KKT equations

$$\begin{pmatrix} \tilde{H}_k & \tilde{J}_k^T \\ \tilde{J}_k & 0 \end{pmatrix} \begin{pmatrix} \tilde{p}_k \\ -\hat{\pi}_k \end{pmatrix} = - \begin{pmatrix} (g_k + H_k \eta_k)_{\mathcal{B}} \\ c_k + J_k \eta_k \end{pmatrix}, \quad (2.20)$$

where η_k is now defined in terms of the final QP nonbasic set, i.e.,

$$(\eta_k)_i = \begin{cases} (\hat{x}_k - x_k)_i & \text{if } i \in \mathcal{N}; \\ 0 & \text{if } i \notin \mathcal{N}. \end{cases} \quad (2.21)$$

As in (2.19), if the basic-nonbasic partition is not changed during the solution of the subproblem, then $\eta_k = 0$. If this final QP nonbasic set is used to define the initial nonbasic set for the next QP subproblem, it is typical for the later QP subproblems to reach optimality in a *single iteration* because the solution of the first QP KKT system satisfies the QP optimality conditions immediately. In this case, the phase-1 procedure simply performs a feasibility check that would be required in any case.

Constraint regularization. One of the purposes of regularization is to define KKT equations that are nonsingular regardless of the rank of \bar{J}_k . Consider the perturbed version of equations (2.19) such that

$$\begin{pmatrix} \bar{H}_k & \bar{J}_k^T \\ \bar{J}_k & -\mu I \end{pmatrix} \begin{pmatrix} \bar{p}_k \\ -\hat{\pi}_k \end{pmatrix} = - \begin{pmatrix} (g_k + H_k \eta_k)_x \\ c_k + J_k \eta_k \end{pmatrix}, \quad (2.22)$$

where μ is a small positive constant. In addition, assume that $\bar{Z}_k^T \bar{H}_k \bar{Z}_k$ is positive definite, where the columns of \bar{Z}_k form a basis for the null space of \bar{J}_k . With this assumption, the unperturbed KKT equations (2.19) are singular if and only if \bar{J}_k has linearly dependent rows.

For simplicity, assume that $\eta_k = 0$. Let $(U \ V)$ be an orthonormal matrix such that the columns of U form a basis for $\text{null}(\bar{J}_k^T)$ and the columns of V form a basis for $\text{range}(\bar{J}_k)$. The unique expansion $\hat{\pi}_k = U\pi_U + V\pi_V$ allows us to rewrite (2.22) as

$$\begin{pmatrix} \bar{H}_k & \bar{J}_k^T V & \\ V^T \bar{J}_k & -\mu I & \\ & & -\mu I \end{pmatrix} \begin{pmatrix} \bar{p}_k \\ -\pi_V \\ -\pi_U \end{pmatrix} = - \begin{pmatrix} (g_k)_x \\ V^T c_k \\ 0 \end{pmatrix}, \quad (2.23)$$

where $\bar{J}_k^T U = 0$ from the definition of U , and $U^T c_k = 0$ because $c_k \in \text{range}(\bar{J}_k)$. The following simple argument shows that the equations (2.23) are nonsingular, regardless of the rank of \bar{J}_k . First, observe that $V^T \bar{J}_k$ has full row rank. Otherwise, if $v^T V^T \bar{J}_k = 0$, it must be the case that $Vv \in \text{null}(\bar{J}_k^T)$. But since $Vv \in \text{range}(V)$ and $\text{range}(V)$ is orthogonal to $\text{null}(\bar{J}_k^T)$, we conclude that $Vv = 0$, and the linearly independence of the columns of V gives $v = 0$.

Moreover, equations (2.23) imply that $\pi_U = 0$ and $\hat{\pi}_k \in \text{range}(\bar{J}_k)$. If $(g_{k+1})_x$ denotes the free components of $g_{k+1} = g_k + H p_k$, then

$$\bar{J}_k^T \hat{\pi}_k = (g_{k+1})_x \quad \text{and} \quad \hat{\pi}_k \in \text{range}(\bar{J}_k).$$

These are the necessary and sufficient conditions for $\hat{\pi}_k$ to be the unique least-length solution of the compatible equations $\bar{J}_k^T \pi = (g_{k+1})_x$. This implies that the regularization gives a unique vector of multipliers.

Wright [173, 174, 175] and Hager [117] show that an SQP method using the regularized equations (2.22) will converge at a superlinear rate, even in the degenerate case. In Section A.3 of the Appendix, QP methods are discussed that give equations of the form (2.22) at every outer iteration, not just in the neighborhood of the solution. These methods implicitly shift the constraints by an amount of order μ and give QP multipliers that converge to an $O(\mu)$ estimate of the least-length multipliers.

A related regularization scheme has been proposed and analyzed by Fischer [58], who solves a second QP to obtain the multiplier estimates. Anitescu [3] regularizes the problem by imposing a trust-region constraint on the plain SQP subproblem (2.1) and solving the resulting subproblem by a semidefinite programming method.

3. The formulation of modern SQP methods. SQP methods have evolved considerably since Wilson's thesis appeared in 1963. Current

implementations of SQP methods for large-scale optimization have solved problems with as many as 40,000 variables and inequality constraints (see, e.g., Gill, Murray and Saunders [83]). During this evolution, both the theory and practice of SQP methods have benefited substantially from developments in competing methods. Similarly, research in SQP methods has had a considerable impact on the formulation and analysis of rival methods—for example, on the treatment of equality constraints in interior methods. On the surface, many recent SQP methods bear little resemblance to the plain SQP method proposed by Wilson. In this section we review some of the principal developments in SQP methods since 1963 while emphasizing connections to other methods. In our discussion, we use the broad definition of an SQP method as one that uses a quadratic programming subproblem to estimate the active set. Implicit in this definition is the assumption that, in the neighborhood of the solution, an SQP method will solve the Newton KKT equations (or some approximation) defined in terms of the free variables.

The complex interrelationships that exist between optimization methods make it difficult (and controversial) to give a precise taxonomy of the many different SQP approaches. Instead, we will discuss methods under four topics that, in our opinion, were influential in shaping developments in the area. Each of these topics will provide a starting-point for discussion of related methods and extensions. The topics are: (i) merit functions and the Han-Powell SQP method, (ii) sequential unconstrained methods, (iii) line-search and trust-region filter methods, and (iv) methods that solve a convex program to determine an estimate of the active set. The modern era of SQP methods can be traced to the publication of the Han-Powell method in 1976 [118, 153]. (It may be argued that almost all subsequent developments in SQP methods are based on attempts to correct perceived theoretical and practical deficiencies in the Wilson-Han-Powell approach.) The sequential unconstrained approaches to SQP have evolved from a 1982 paper by Fletcher [61, 62]. Filter SQP methods are a more recent development, being proposed by Fletcher and Leyffer [66, 67] in 1998.

3.1. Review of line-search and trust-region methods. Our discussion of the equality constrained problem in Section 2.1.1 emphasizes the local equivalence between a plain SQP method and Newton’s method applied to the first-order optimality conditions. As the Newton iterates may diverge or may not be well-defined if the starting point is not sufficiently close to a solution, some modification is needed to force convergence from arbitrary starting points. Line-search methods and trust-region methods are two alternative modifications of Newton’s method. We begin by reviewing the main properties of these methods in the context of unconstrained minimization.

3.1.1. Line-search methods: the unconstrained case. Associated with the k th iteration of a conventional line-search method for un-

constrained optimization is a scalar-valued function $m_k(x)$ that represents a *local line-search model* of f . The next iterate is then $x_{k+1} = x_k + d_k$, where d_k is chosen so that the improvement in f is at least as good as a fixed fraction of the improvement in the local model, i.e., d_k must satisfy

$$f(x_k) - f(x_k + d_k) \geq \eta(m_k(x_k) - m_k(x_k + d_k)), \quad (3.1)$$

where η is a fixed parameter such that $0 < \eta < \frac{1}{2}$. Typical line-search models are affine and quadratic functions based on a first- or second-order Taylor-series approximation of f . For example, a first-order approximation provides the affine line-search model $m_k(x) = f(x_k) + g(x_k)^T(x - x_k)$. In a general line-search method, the change in variables has the form $d_k \equiv d_k(\alpha_k)$, where α_k is a scalar steplength that defines a point on the parameterized path $d_k(\alpha)$. In the simplest case, $d_k(\alpha) = \alpha p_k$, where p_k is an approximate solution of the unconstrained subproblem $\min_{p \in \mathbb{R}^n} g_k^T p + \frac{1}{2} p^T B_k p$, with B_k a *positive-definite* approximation of the Hessian H_k . More generally, if H_k is indefinite, $d_k(\alpha)$ is defined in terms of p_k and a direction s_k such that $s_k^T H_k s_k < 0$ (see, e.g., Goldfarb [99], Moré and Sorensen [138], and Olivares, Moguerza and Prieto [144]). A crucial feature of a line-search method is that $d_k(\alpha)$ is defined in terms of a *convex* subproblem, which may be defined implicitly during the calculation of p_k ; see, e.g., Greenstadt [112], Gill and Murray [81], Schnabel and Eskow [163]).

Condition (3.1) may be written as $f(x_k) - f(x_k + d_k) \geq \eta \Delta m_k(d_k)$, where the quantity

$$\Delta m_k(d) = m_k(x_k) - m_k(x_k + d) \quad (3.2)$$

is the change in f predicted by the line-search model function. An essential property of the line-search model is that it must always be possible to find an α_k that satisfies (3.1). In particular, there must exist a positive $\bar{\alpha}$ such that

$$f(x_k + d_k(\alpha)) \leq f(x_k) - \eta \Delta m_k(d_k(\alpha)), \quad \text{for all } \alpha \in (0, \bar{\alpha}). \quad (3.3)$$

For this condition to hold, the model must predict a reduction in $f(x)$ at $x = x_k$, i.e., $\Delta m_k(d_k(\alpha)) > 0$ for all $\alpha \in (0, \bar{\alpha})$. Under the assumption that (3.3) holds, there are various algorithms for finding an appropriate α_k . For example, in a *backtracking line search*, the step $\alpha_k = 1$ is decreased by a fixed factor until condition (3.1) is satisfied. It can be shown that this simple procedure is enough to guarantee a sufficient decrease in f . More sophisticated methods satisfy (3.1) in conjunction with other conditions that ensure a sufficient decrease in f (see, e.g., Ortega and Rheinboldt [145], Moré and Thuente [140], and Gill et al. [86]).

The line-search methods defined above enforce a monotone decrease in f at each iteration. In some cases the definition of $d_k(\alpha)$ may warrant the use of a *nonmonotone* line search in which f is permitted to increase on some iterations. An example of a nonmonotone line-search condition is

$$f(x_k + d_k(\alpha)) \leq \max_{0 \leq j \leq r} [f(x_{k-j})] - \eta \Delta m_k(d_k(\alpha)),$$

where r is some fixed number of previous iterations (for other schemes of varying complexity, see, e.g., Grippo, Lampariello and Lucidi [113, 114, 115], Toint [167], and Zhang and Hager [180]). In Section 3.2, we discuss the “watchdog technique”, which is a nonmonotone line search that allows the value $\alpha_k = 1$ to be used for a limited number of steps, regardless of the value of f .

3.1.2. Trust-region methods: the unconstrained case. When there are no constraints, line-search methods and trust-region methods have many properties in common. Both methods choose the value of a scalar variable so that the objective improves by an amount that is at least as good as a fraction of the improvement in a local model (see condition (3.1)). A crucial difference is that a line-search method involves the solution of a bounded *convex* subproblem. By contrast, trust-region methods solve a constrained, possibly nonconvex, subproblem of the form

$$\min_{d \in \mathbb{R}^n} g_k^T d + \frac{1}{2} d^T H_k d \quad \text{subject to } \|d\| \leq \delta_k, \quad (3.4)$$

with condition (3.1) being enforced, if necessary, by reducing the positive scalar δ_k (the trust-region radius). The final value of δ_k is also used to define an initial estimate of δ_{k+1} , with the possibility that δ_{k+1} is increased to a multiple of δ_k if the reduction in f is significantly better than the reduction predicted by the model. If the trust-region radius is reduced over a sequence of consecutive iterations, the step d_k will go to zero along the direction of steepest descent with respect to the particular norm used to define the trust region. As in a line search, it is possible to define trust-region methods that do not enforce a reduction in f at every step (see, e.g., Gu and Mo [116]).

The complexity of constrained minimization is generally higher than that of unconstrained minimization. Moreover, the trust-region subproblem may need to be solved more than once before the condition (3.1) is satisfied. Nevertheless, trust-region methods provide computational benefits when some of the eigenvalues of H_k are close to zero (see Kroyan [126]). Modern trust-region methods require only an approximate solution of (3.4). For a comprehensive review of trust-region methods for both unconstrained and constrained optimization, see Conn, Gould and Toint [42].

3.2. The Han-Powell method. Han [119] and Powell [153] introduced two crucial improvements to the plain SQP method of Wilson. The first was the use of a QP subproblem defined in terms of a *positive-definite* quasi-Newton approximation. The second was the use of a *line-search merit function* to obtain a sequence of improving estimates of the solution.

A merit function \mathcal{M} is a scalar-valued function whose value provides a measure of the quality of a given point as an estimate of a solution of the constrained problem. Each value of \mathcal{M} represents a compromise between

the (usually conflicting) aims of minimizing the objective function and minimizing the constraint violations. Analogous to the unconstrained case, the merit function is used in conjunction with a line-search model $m_k(x)$ to define a sufficient decrease at the k th iteration. In the constrained case, d_k is chosen to satisfy

$$\mathcal{M}(x_k) - \mathcal{M}(x_k + d_k) \geq \eta(m_k(x_k) - m_k(x_k + d_k)), \quad x_k + d_k \geq 0. \quad (3.5)$$

Han and Powell proposed the use of the ℓ_1 penalty function (1.8) as a merit function, i.e., $\mathcal{M}(x) \triangleq \mathcal{M}(x; \rho) = P_1(x; \rho)$. Moreover, they suggested that $d_k(\alpha) = \alpha p_k = \alpha(\hat{x}_k - x_k)$, where \hat{x}_k is the solution of the convex subproblem

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T B_k(x - x_k) \\ & \text{subject to} \quad c_k + J_k(x - x_k) = 0, \quad x \geq 0, \end{aligned} \quad (3.6)$$

with B_k a positive-definite approximation to the Hessian of the Lagrangian or augmented Lagrangian (in Section 3.2.1 below, we discuss the definition of B_k in this context). As the QP (3.6) is convex, it may be solved using either a primal or dual active-set method (see Section A.1 of the Appendix). In either case, the QP multipliers $\hat{\pi}_k$, and vector \tilde{p}_k of components of p_k in the final basic set satisfy the nonsingular KKT equation

$$\begin{pmatrix} \tilde{B}_k & \tilde{J}_k^T \\ \tilde{J}_k & \end{pmatrix} \begin{pmatrix} \tilde{p}_k \\ -\hat{\pi}_k \end{pmatrix} = - \begin{pmatrix} (g_k + B_k \eta_k)_B \\ c_k + J_k \eta_k \end{pmatrix}, \quad (3.7)$$

where \tilde{B}_k and \tilde{J}_k denote the matrices of basic components of B_k and J_k and η_k is defined as in (2.21).

3.2.1. Quasi-Newton approximations. Many methods for unconstrained minimization use a quasi-Newton approximation of the Hessian when second derivatives are either unavailable or too expensive to evaluate. Arguably, the most commonly used quasi-Newton approximation is defined using the BFGS method (see Broyden [14], Fletcher[59], Goldfarb[98], and Shanno [164]). Given iterates x_k and x_{k+1} , and a symmetric approximate Hessian B_k , the BFGS approximation for the next iteration has the form

$$B_{k+1} = B_k - \frac{1}{d_k^T B_k d_k} B_k d_k d_k^T B_k + \frac{1}{y_k^T d_k} y_k y_k^T, \quad (3.8)$$

where $d_k = x_{k+1} - x_k$ and $y_k = g(x_{k+1}) - g(x_k)$. If B_k is positive definite, then B_{k+1} is positive definite if and only if the approximate curvature $y_k^T d_k$ is positive.

For the constrained case, Han [119] proposed maintaining a BFGS approximation of the Hessian of the augmented Lagrangian function

$$L_A(x, \pi; \rho) = f(x) - c(x)^T \pi + \frac{1}{2} \rho c(x)^T c(x).$$

(As the Hessian of the Lagrangian does not include the linear constraints, we have omitted z from the Lagrangian term.) This implies that the gradient difference y_k in (3.8) involves the gradient of the augmented Lagrangian, with

$$d_k = x_{k+1} - x_k, \quad \text{and} \quad y_k = \nabla_x L_A(x_{k+1}, \pi_{k+1}; \rho) - \nabla_x L_A(x_k, \pi_{k+1}; \rho),$$

where π_{k+1} are estimates of the optimal dual variables. This proposal is motivated by the fact that if ρ is sufficiently large, the Hessian of $L(x, \pi; \rho)$ is positive definite for all (x, π) close to an isolated solution (x^*, π^*) (see also, Tapia [165], and Byrd, Tapia and Zhang [29]).

The use of an augmented Lagrangian Hessian for the QP subproblem changes the properties of the QP dual variables. In particular, if $(\hat{x}_k, \hat{\pi}_k, \hat{z}_k)$ is the solution of the QP (3.6) with B_k defined as $H_k + \rho J_k^T J_k$, then $(\hat{x}_k, \hat{\pi}_k + \rho c_k, \hat{z}_k)$ is the solution of the QP (3.6) with B_k replaced by H_k (assuming that the same local solution is found when H_k is not positive definite). In other words, if the augmented Lagrangian Hessian is used instead of the Lagrangian Hessian, the x and z variables do not change, but the π -values are shifted by ρc_k . An appropriate value for π_{k+1} in the definition of y_k is then $\pi_{k+1} = \hat{\pi}_k + \rho c_k$, giving, after some simplification,

$$y_k = g_{k+1} - g_k - (J_{k+1} - J_k)^T \hat{\pi}_k + \rho J_{k+1}^T (c_{k+1} - c_k).$$

If the approximate curvature $y_k^T d_k$ is not positive, the matrix B_{k+1} of (3.8) is either indefinite or undefined. In terms of an update to the Hessian of the augmented Lagrangian, a negative $y_k^T d_k$ implies that either ρ is not sufficiently large, or the curvature of the penalty term $\frac{1}{2}\rho c(x)^T c(x)$ is negative along d_k . In the first case, ρ must be increased by an amount that is sufficiently large to give a positive value of $y_k^T d_k$. In the second case, the approximate curvature of the Lagrangian is not sufficiently positive and there is no finite ρ that gives $y_k^T d_k > 0$. In this case, the update should be skipped. The curvature is considered not sufficiently positive if

$$y_k^T d_k < \sigma_k, \quad \sigma_k = \alpha_k (1 - \eta) p_k^T B_k p_k, \quad (3.9)$$

where η is a preassigned constant ($0 < \eta < 1$) and p_k is the search direction $\hat{x}_k - x_k$ defined by the QP subproblem. If $y_k^T d_k < \sigma_k$, then ρ is replaced by $\rho + \Delta\rho$, where

$$\Delta\rho = \begin{cases} \frac{\sigma_k - y_k^T d_k}{d_k^T J_{k+1}^T (c_{k+1} - c_k)}, & \text{if } d_k^T J_{k+1}^T (c_{k+1} - c_k) > 0; \\ 0, & \text{otherwise.} \end{cases}$$

If $\Delta\rho = 0$, the approximate curvature of $c(x)^T c(x)$ is not positive and the update should be skipped.

Maintaining an approximation of the Hessian of $L_A(x; \pi, \rho)$ involves a number of difficulties, all of which stem from the need to increase the value

of ρ . First, the usual convergence of the sequence $\{B_k\}$ is disrupted when ρ is increased. Second, a large increase in ρ will give an ill-conditioned matrix B_{k+1} . Finally, because ρ is always increased, the ill-effects of large values of ρ persist throughout the computation.

Powell [153] suggested the use of a positive-definite BFGS approximation for the *Lagrangian* Hessian, i.e., the update pair is

$$\begin{aligned} d_k &= x_{k+1} - x_k, \\ y_k &= \nabla_x L(x_{k+1}, \pi_{k+1}, z_{k+1}) - \nabla_x L(x_k, \pi_{k+1}, z_{k+1}). \end{aligned} \quad (3.10)$$

If the QP multipliers are used for π_{k+1} , the difference in Lagrangian gradients is given by $y_k = g_{k+1} - g_k - (J_{k+1} - J_k)^T \hat{\pi}_k$.

A *positive-definite* BFGS approximation may appear to be a surprising choice for B_k , given that the Hessian of the Lagrangian is generally *indefinite* at the solution. However, Powell's proposal is based on the observation that the approximate curvature is likely to be positive in the neighborhood of an isolated solution, even when the Hessian of the Lagrangian is indefinite. The reason for this is that the iterates of a quasi-Newton SQP converge to the solution along a path that lies in the null space of the "free" columns of the Jacobian. As the Lagrangian Hessian is generally positive definite along this path, the approximate curvature $y_k^T d_k$ is positive as the iterates converge and an R-superlinear convergence rate is obtained. Powell's proposal may be justified by considering the properties of $(\hat{x}_k, \hat{\pi}_k)$, the solution of the QP subproblem. Let $p_k = \hat{x}_k - x_k$ and $\tilde{g}(x) = g_k + B_k(x - x_k)$. It is shown in Section A.4.1 of the Appendix that $(\hat{x}_k, \hat{\pi}_k)$ satisfies the equations

$$\begin{aligned} U_k p_Y &= -c_k, & p_N &= Y_k p_Y, \\ x_F = x_k + p_N, \quad Z_k^T B_k Z_k p_Z &= -Z_k^T \tilde{g}(x_F), & p_T &= Z_k p_Z, \\ p_k = p_N + p_T, \quad U_k^T \hat{\pi}_k &= Y_k^T \tilde{g}(x_k + p_k), \end{aligned} \quad (3.11)$$

where U_k is nonsingular and the columns of Z_k lie in the null space of J_k . These equations indicate that the QP step is the sum of the vectors p_N and p_T , where p_N is the Newton step to the linearized constraints and p_T is a quasi-Newton step based on *approximate* second-derivative information associated with the reduced Hessian $Z_k^T B_k Z_k$. Because of this disparity in the quality of the Newton steps, *the constraints tend to converge to zero faster than the reduced gradient* and the convergence of a quasi-Newton SQP method is characterized by the relationship $\|p_N\|/\|p_T\| \rightarrow 0$, i.e., the final search directions lie almost wholly in the null space of $J(x^*)$.

If x_k is far from a solution, the approximate curvature $y_k^T d_k$ may not be positive and the formula (3.8) will give an indefinite or undefined B_{k+1} . If, as in the case of unconstrained minimization, the update is skipped when $y_k^T d_k \leq 0$, no new information about curvature of the Lagrangian will be gained. In this situation, an alternative pair of vectors satisfying

$y_k^T d_k > 0$ can be used. Given the definition (3.9) of the least permissible approximate curvature, Powell [152] redefines y_k as $y_k + \Delta y_k$, where Δy_k chosen so that $(y_k + \Delta y_k)^T d_k = \sigma_k$, i.e.,

$$\Delta y_k = \frac{\sigma_k - y_k^T d_k}{d_k^T (y_k - B_k d_k)} (y_k - B_k d_k).$$

The Powell modification is always well defined, which implies that it is always applied—even when it might be unwarranted because of negative curvature of the Lagrangian in the null space of \tilde{J}_k (cf. (3.7)).

3.2.2. Properties of the merit function. The Han-Powell merit function $\mathcal{M}(x; \rho) = P_1(x; \rho)$ has the appealing property that x^* is an unconstrained minimizer of $P_1(x; \rho)$ for $\rho > \|\pi^*\|_\infty$ (see, e.g., Zangwill [179], and Han and Mangasarian [120]). A potential line-search model for $P_1(x; \rho)$ is

$$m_k(x; \rho) = f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T B_k(x - x_k) + \rho \|c_k + J_k(x - x_k)\|_1,$$

which is the ℓ_1 penalty function defined with local affine and quadratic approximations for c and f . However, because B_k is positive definite, a stronger condition on α is defined by omitting the quadratic term and using the line-search model

$$m_k(x; \rho) = f_k + g_k^T(x - x_k) + \rho \|c_k + J_k(x - x_k)\|_1. \quad (3.12)$$

To obtain a smaller value of $P_1(x; \rho)$ at each iteration, the line-search model must satisfy $\Delta m_k(d_k; \rho) > 0$, where $\Delta m_k(d; \rho)$ is the predicted reduction in \mathcal{M} analogous to (3.2). The optimality conditions (2.18) for the QP subproblem together with the affine model (3.12) defined with $x = x_k + \alpha p_k$ allow us to write the predicted reduction as

$$\begin{aligned} \Delta m_k(\alpha p_k; \rho) &= \alpha (\rho \|c_k\|_1 + c_k^T \hat{\pi}_k - p_k^T \hat{z}_k + p_k^T B_k p_k) \\ &= \alpha (\rho \|c_k\|_1 + c_k^T \hat{\pi}_k - (\hat{x}_k - x_k)^T \hat{z}_k + p_k^T B_k p_k). \end{aligned} \quad (3.13)$$

The QP optimality conditions give $\hat{x}_k \cdot \hat{z}_k = 0$, yielding

$$\begin{aligned} \Delta m_k(\alpha p_k; \rho) &= \alpha (\rho \|c_k\|_1 + c_k^T \hat{\pi}_k + x_k^T \hat{z}_k + p_k^T B_k p_k) \\ &\geq \alpha \left(\sum_{i=1}^m |c_i(x_k)| (\rho - |\hat{\pi}_k|_i) + \|x_k \cdot \hat{z}_k\|_1 + p_k^T B_k p_k \right), \end{aligned}$$

which implies that if B_k is positive definite, then a sufficient condition for $\Delta m_k(\alpha p_k; \rho) > 0$ is $\rho \geq \|\hat{\pi}_k\|_\infty$. Han [118] uses this condition to define a nondecreasing sequence $\{\rho_k\}$ such that $\rho_k > \|\hat{\pi}_j\|_\infty$ for all $k \geq j$. With this definition of $\{\rho_k\}$, and under assumptions that include the uniform boundedness of the sequence $\{B_k\}$ and the existence of at least one nonnegative x such that $c_k + J_k(x - x_k) = 0$, Han shows that all accumulation points of the sequence $\{x_k\}$ are first-order KKT points of the constrained problem (1.2).

3.2.3. Extensions. The introduction of the Wilson-Han-Powell SQP method (i.e., the plain SQP method with a convex subproblem and a line-search with respect to a merit function) had an immediate beneficial effect on the performance of optimization codes. However, as is the case with all successful innovations, it was not long before certain issues were identified that have an impact on performance and reliability. In this section we consider some of these issues and outline some extensions of the Wilson-Han-Powell method that are intended to address them.

The Maratos effect and alternative merit functions. The value of the penalty parameter in the ℓ_1 merit function \mathcal{M} in (3.5) can have a substantial effect on the overall efficiency of an SQP method. When solving a sequence of related problems, it may be possible to provide a good estimate of the optimal multipliers, and a value of $\rho \approx \|\pi^*\|_\infty$ can be specified. When ρ is large relative to the magnitude of f , the level surfaces of \mathcal{M} closely resemble the constraint surface $c(x) = 0$. If the constraints are changing rapidly and the SQP outer iterates become close to a nonoptimal point near the constraints (as is the case for methods that use a quasi-Newton approximation B_k , see Section 3.2.1), the iterates must negotiate the base of a steep-sided curved valley. In this situation, the affine model of the constraints provides for only a limited amount of progress along the SQP direction, and the step $\alpha = 1$ fails to reduce the value of \mathcal{M} . This rejection of the plain SQP step near x^* causes a breakdown of the superlinear convergence rate. Various strategies have been devised to prevent this phenomenon, which is known as the “Maratos effect” (see Maratos [134]). One approach is to use a “nonmonotone” line search that allows the merit function to increase for a limited number of iterations (see, e.g., Chamberlain et al. [30], and Dai and Schittkowski [45]).

Another approach, proposed by Fletcher [62], seeks to reduce the magnitude of the penalty term by computing a *second-order correction* to the Newton step. The second-order correction uses the step $p_k + s_k$, where the step s_k is the solution of a second subproblem:

$$\begin{aligned} & \underset{s \in \mathbb{R}^n}{\text{minimize}} \quad f_k + g_k^T(p_k + s) + \frac{1}{2}(p_k + s)^T B_k(p_k + s) \\ & \text{subject to} \quad c(x_k + p_k) + J_k s = 0, \quad x_k + p_k + s \geq 0. \end{aligned} \tag{3.14}$$

The second-order correction requires an additional constraint evaluation at the SQP point $x_k + p_k$.

If a feasible-point active-set method is used to solve (3.6) and the subproblem is started with the basic set from the previous iteration, the second-order correction need be computed only if (3.6) is solved in one iteration, which is a necessary condition for the method to be in the final stages of convergence. If the solution of (3.14) is also identified in one iteration and \tilde{J}_k is the matrix of basic columns of J_k , then s_k satisfies the equations

$$\begin{pmatrix} \tilde{B}_k & \tilde{J}_k^T \\ \tilde{J}_k & 0 \end{pmatrix} \begin{pmatrix} s_k \\ -\bar{\pi}_k \end{pmatrix} = - \begin{pmatrix} (g_k + B_k(p_k + \eta_k))_B \\ c(x_k + p_k) + J_k \eta_k \end{pmatrix}, \quad (3.15)$$

where $\bar{\pi}_k$ is the vector of optimal multipliers for (3.14) and η_k is defined as in (2.21). In this situation, if some factorization of the KKT matrix is available on termination of the solution of (3.6), the correction may be obtained with just one solve with a different right-hand side. For an analysis of the rate of convergence, see Fletcher [62] and Yuan [177].

Other merit functions may be defined for different choices of the norm of the constraint violations. For the infinity-norm, the ℓ_∞ penalty function $P_\infty(x; \rho)$ defined in (1.9) may be used in conjunction with the line-search model

$$m_k(x; \rho) = f_k + g_k^T(x - x_k) + \rho \|c_k + J_k(x - x_k)\|_\infty.$$

This model predicts a reduction in $P_\infty(x; \rho)$ if $p_k^T B_k p_k \geq 0$ and $\rho > \|\hat{\pi}_k\|_1$. Anitescu [2] considers the convergence of the ℓ_∞ merit function when applied with various line-search strategies and a convex QP subproblem.

Like its ℓ_1 counterpart, the ℓ_∞ penalty function can exhibit the Maratos effect for large values of ρ . Merit functions that do not have this problem may be defined by using a smooth norm for the constraint violations. In general, a merit function may be defined in terms of the primal variables only, or may include estimates of the Lagrange multipliers. A merit function that does not suffer from the Maratos effect is the augmented Lagrangian function:

$$\mathcal{M}(x, \pi; \rho) \equiv f(x) - \pi^T c(x) + \frac{1}{2} \rho c(x)^T c(x), \quad (3.16)$$

where π is a multiplier estimate and ρ is a nonnegative penalty parameter. Schittkowski [160, 161, 162], and Gill et al. [93, 83] define SQP methods in which both the primal and dual variables are modified by the line search, with

$$x_{k+1} = x_k + \alpha_k p_k, \quad \pi_{k+1} = \pi_k + \alpha_k q_k, \quad (3.17)$$

where the primal-dual search directions $p_k = \hat{x}_k - x_k$ and $q_k = \hat{\pi}_k - \pi_k$ are based on the solution $(\hat{x}_k, \hat{\pi}_k)$ of a convex QP with a quasi-Newton Hessian. When an augmented Lagrangian is used in the conventional role as an objective function for sequential unconstrained minimization, new multiplier estimates are obtained by *maximizing* with respect to the dual variables. In the SQP context, the inclusion of the dual variables as arguments for *minimization* serves to make the augmented Lagrangian a continuous function of both the primal and dual variables, with the step length acting as a continuation parameter that links the old and new values of π . If necessary, the penalty parameter is increased to ensure that the primal-dual direction is a descent direction for the merit function. However, it can be shown that

under typical assumptions on the problem, the penalty parameter remains bounded (see Gill et al. [93], and Murray and Prieto [142] for details). If the objective is convex and the feasible region is a convex set, it is often the case that the penalty parameter never needs to be increased from an initial value of zero.

A number of line-search SQP methods have been proposed that use variants of the conventional augmented Lagrangian as a merit function (see, e.g., DiPillo and Grippo [49], Bertsekas [6], Byrd, Tapia and Zhang [29], and Anitescu [2]). A primal-dual augmented Lagrangian has been proposed by Gill and Robinson [95]. Given an estimate π_E of the multipliers π^* , consider the function

$$\begin{aligned} L_A(x, \pi; \pi_E, \mu) = f(x) - c(x)^T \pi_E + \frac{1}{2\mu} \|c(x)\|_2^2 \\ + \frac{1}{2\mu} \|c(x) + \mu(\pi - \pi_E)\|^2, \end{aligned} \quad (3.18)$$

where μ is a positive inverse penalty parameter (see also, Forsgren and Gill [71], Robinson [155], and Gill and Robinson [95]). The primal-dual augmented Lagrangian has a bound-constrained minimization property analogous to the conventional augmented Lagrangian (1.11). In particular, if π_E is given the value of the optimal multiplier vector π^* , then (x^*, π^*) is a first-order KKT point for the bound-constrained problem

$$\underset{x \in \mathbb{R}^n; \pi \in \mathbb{R}^m}{\text{minimize}} \quad L_A(x, \pi; \pi^*, \mu) \quad \text{subject to } x \geq 0.$$

Moreover, if the second-order sufficient conditions for optimality hold, then there exists a finite $\bar{\mu}$ such that (x^*, π^*) is an isolated unconstrained minimizer of L_A for all $\mu < \bar{\mu}$. It follows that L_A may be minimized simultaneously with respect to both the primal and dual variables. A benefit of using L_A as an SQP merit function is that it may be used in conjunction with a regularized method for solving the QP subproblem (see Section A.3 of the Appendix for details).

We conclude this section by mentioning methods that avoid the need for a merit function altogether by generating iterates that are always feasible. In many physical and engineering applications, the constraint functions not only characterize the desired properties of the solution, but also define a region in which the problem statement is meaningful (for example, $f(x)$ or some of the constraint functions may be undefined outside the feasible region). In these applications, an interior point can usually be determined trivially. Interior methods are therefore highly appropriate for this class of problem. However, several SQP methods have been proposed for optimization in this context, see, e.g., Lawrence and Tits [127], and Kostreva and Chen [124, 125]. These methods are suitable for problems that have only inequality constraints, the only exception being *linear* equality con-

straints, which can be kept feasible at every iterate (see, e.g., Gill, Murray and Wright [94]).

Formulation of the QP subproblem. A potential difficulty associated with SQP methods based on the direct linearization of the nonlinear constraints, is that the QP subproblem may be infeasible. This can be caused by the nonlinear constraints being infeasible, or by a poor linearization at the current iterate. In the context of the Wilson-Han-Powell method, this problem may be resolved by perturbing the QP subproblem so that the constraints always have a feasible point. The magnitude of the perturbation is then reduced or eliminated as the iterates converge. Powell [153] focused on the case of an infeasible linearization and considered the modified QP:

$$\begin{aligned} \underset{x \in \mathbb{R}^n, \theta \in \mathbb{R}^1}{\text{minimize}} \quad & f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T B_k(x - x_k) + \frac{1}{2}\rho_k(1 - \theta)^2 \\ \text{subject to} \quad & (1 - \theta)c_k + J_k(x - x_k) = 0, \quad x + \theta[x_k]_- \geq 0, \end{aligned} \quad (3.19)$$

where $[x]_-$ is the vector with components $\max\{-x_i, 0\}$, and θ is an additional variable that is driven to zero by increasing the nonnegative penalty parameter ρ_k . The modified QP is always feasible—for example, the point $(x, \theta) = (x_k, 1)$ satisfies the constraints.

Burke [21, 22], and Burke and Han [23] use a different approach that is based on the observation that the problem (1.2) is actually two problems in one: the *feasibility problem* of satisfying the constraints, and the *optimality problem* of minimizing f . They define a line-search algorithm that has the primary goal of attaining feasibility.

The computation of the line-search direction is organized into two phases. The first phase ignores the objective and computes a descent direction for a function that measures the distance of an arbitrary point to the set of feasible points for the nonlinear problem. The required direction is computed by minimizing the distance to the feasible set for the *linearized* constraints. For the second phase, the constraint residuals corresponding to the optimal value of the distance function are used to modify the constraints of the conventional QP subproblem. The modified QP is always feasible, and the resulting direction is used in a line search with a merit function that includes a term involving the value of the distance function. Under certain assumptions, this procedure provides a sequence that converges to a first-order stationary point of either the original problem or the distance function.

The definition of the distance function requires a choice of norm, although Burke and Han provide a general analysis that is independent of the norm. For simplicity, we describe the computations for each phase when the distance function is defined in terms of the one-norm. Given current values of parameters σ_k and β_k such that $0 < \sigma_k \leq \beta_k$, the first phase involves the solution of the linear program

$$\begin{aligned}
& \underset{x, v \in \mathbb{R}^n; u \in \mathbb{R}^m}{\text{minimize}} && e^T u + e^T v \\
& \text{subject to} && -u \leq c_k + J_k(x - x_k) \leq u, \quad x + v \geq 0, \quad v \geq 0, \\
& && -\sigma_k e \leq x - x_k \leq \sigma_k e.
\end{aligned} \tag{3.20}$$

This problem gives vectors u and v of least one-norm for which the constraints $c_k + J_k(x - x_k) = u$, $x + v \geq 0$ and $\|x - x_k\|_\infty \leq \sigma_k$ are feasible. If the original linearized constraints are feasible, then the work necessary to solve problem (3.20) is comparable to that of the feasibility phase of a two-phase active-set method for the plain QP subproblem (see Section A.1). The difference is the extra expense of locating a bounded feasible point with *least-length* distance from x_k . Let x_F denote the computed solution of the phase-1 problem (3.20). The computation for phase 2 involves the solution of the QP:

$$\begin{aligned}
& \underset{x \in \mathbb{R}^n}{\text{minimize}} && f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T B_k(x - x_k) \\
& \text{subject to} && c_k + J_k(x - x_k) = \hat{c}_k(x_F), \quad x + [x_F]_- \geq 0, \\
& && -\beta_k e \leq x - x_k \leq \beta_k e,
\end{aligned} \tag{3.21}$$

where, as usual, $\hat{c}_k(x)$ denotes the vector of linearized constraint functions $\hat{c}_k(x) = c_k + J_k(x - x_k)$. The phase-2 problem (3.21) is a convex program with bounded solution \hat{x}_k (say). This solution is used to define the search direction $p_k = \hat{x}_k - x_k$ for a line search on the merit function

$$\mathcal{M}(x) = f(x) + \rho_k \|c(x)\|_1 + \rho_k \| [x]_- \|_1.$$

Once $x_{k+1} = x_k + \alpha_k p_k$ has been determined, the positive-definite approximate Hessian B_k is updated as in the conventional Han-Powell method. For details on how the parameters ρ_k , σ_k and β_k are updated, the reader is referred to Burke and Han [23]. Other related variants of the Wilson-Han-Powell formulation are proposed by Liu and Yuan [130], and Mo, Zhang and Wei [135].

Another approach to the treatment of infeasibility is to modify the original *nonlinear* constraints so that the linearized constraints of the QP subproblem are always feasible. This is the approach taken in the method of SNOPT (see Gill, Murray and Saunders [83]). The method proceeds to solve (1.2) as given, using QP subproblems based on the conventional linearization of the nonlinear constraints. If a QP subproblem proves to be infeasible or unbounded (or if the Lagrange multiplier estimates become large), SNOPT enters *nonlinear elastic mode* and switches to the nonlinear elastic problem (1.4). The QP subproblem for the nonlinear elastic problem is given by

$$\begin{aligned}
& \underset{x \in \mathbb{R}^n; u, v \in \mathbb{R}^m}{\text{minimize}} && \hat{f}_k(x) + \rho_k e^T u + \rho_k e^T v \\
& \text{subject to} && c_k + J_k(x - x_k) - u + v = 0, \\
& && x \geq 0, \quad u \geq 0, \quad v \geq 0,
\end{aligned} \tag{3.22}$$

where $\widehat{f}_k(x) + \rho_k e^T u + \rho_k e^T v$ is the *composite QP objective* and ρ_k is a nonnegative *penalty parameter* or *elastic weight* analogous to the quantity defined in Section 1.3. This problem is always feasible and the solution is used in conjunction with a merit function defined in terms of the nonlinear elastic problem (1.4).

Quasi-Newton updates and indefiniteness. A necessary condition for Q-superlinear convergence is that the approximate Hessian matrices $\{B_k\}$ satisfy

$$\lim_{k \rightarrow \infty} \frac{\|Z_k Z_k^T (B_k - H(x^*, \pi^*)) Z_k Z_k^T d_k\|}{\|d_k\|} = 0,$$

where Z_k is the matrix defined in (3.11) (see Boggs and Tolle [9]). The definition of y_k and d_k should ensure that this condition is satisfied as the solution is approached, so that *Q*-superlinear convergence is not inhibited.

One possible modification uses the intermediate point x_F defined by equations (3.11). If x_F is known, new values of d_k and y_k are computed based on evaluating the nonlinear functions at the point $w_k = x_k + \alpha_k(x_F - x_k)$. The BFGS update is then attempted using the update pair:

$$d_k = x_{k+1} - w_k, \quad y_k = \nabla_x L(x_{k+1}, \pi_{k+1}, z_{k+1}) - \nabla_x L(w_k, \pi_{k+1}, z_{k+1}).$$

The purpose of this modification is to exploit the properties of the reduced Hessian in the neighborhood of a local minimizer of (2.4). With this choice of w_k , the change in variables is $d_k = x_{k+1} - w_k = \alpha_k p_T$, where p_T is the vector $\widehat{x}_k - x_F$ (see (3.11) above). Then,

$$y_k^T d_k = \alpha_k y_k^T p_T \approx \alpha_k^2 p_T^T H(x_k, \pi_{k+1}) p_T = \alpha_k^2 p_z^T Z_k^T H(w_k, \pi_{k+1}) Z_k p_z.$$

It follows that $y_k^T d_k$ approximates the curvature of the reduced Hessian, which is positive definite sufficiently close to an isolated local minimizer of (2.4). If this modification does not provide sufficiently positive approximate curvature, no update is made. An additional function evaluation is required at w_k , but the modification is rarely needed more than a few times—even when the Hessian of the Lagrangian has negative eigenvalues at a solution. (For further information, see Gill, Murray and Saunders [83].)

Large-scale Hessians. If the number of variables is large, conventional quasi-Newton methods are prohibitively expensive because of the need to store the (dense) matrix B_k . A limited-memory approach uses a fixed number of vectors, say ℓ , to define a positive-definite approximation to $H(x_k, \pi_k)$ based on curvature information accrued during the most recent ℓ iterations. Let ℓ be preassigned (say $\ell = 10$), and consider any iteration k such that $k \geq \ell - 1$. Given any initial positive-definite approximation $B_k^{(0)}$ to $H(x_k, \pi_k)$, consider the sequence of matrices $\{B_k^{(i)}\}$, for $i = k - \ell, k - \ell + 1, \dots, k$, such that

$$B_k^{(k-\ell)} = B_k^{(0)}, \quad B_k^{(i+1)} = B_k^{(i)} + v_i v_i^T - u_i u_i^T, \quad i = k - \ell, \dots, k - 1,$$

where the $\{(u_i, v_i)\}$ are ℓ vector pairs with each (u_i, v_i) defined in terms of $(d_{k-\ell}, y_{k-\ell}), \dots, (d_{i-1}, y_{i-1})$ (cf. (3.10)) via

$$u_i = \frac{1}{(d_i^T B_k^{(i)} d_i)^{\frac{1}{2}}} B_k^{(i)} d_i, \quad \text{and} \quad v_i = \frac{1}{(y_i^T d_i)^{\frac{1}{2}}} y_i.$$

Similar limited-memory quasi-Newton approximations are described by Nocedal and Wright [143], Buckley and LeNir [15, 16] and Gilbert and Lemaréchal [77]. More elaborate schemes are given by Liu and Nocedal [129], Byrd, Nocedal, and Schnabel [27], and Gill and Leonard [80], and some have been evaluated by Morales [136].

The definition of B_k requires the ℓ pairs (u_i, v_i) . Each of the vectors u_i ($k-\ell \leq i \leq k-1$) involves the product $B_k^{(i)} d_i$, which is computed using the recurrence relation

$$B_k^{(i)} d_i = B_k^{(0)} d_i + \sum_{j=k-\ell}^{i-1} ((v_j^T d_i) v_j - (u_j^T d_i) u_j).$$

For the vectors v_i ($k-\ell \leq i \leq k-1$) and scalars $v_j^T d_i$ ($k-\ell \leq j \leq i-1$), only v_{k-1} and $v_j^T d_{k-1}$ ($k-\ell \leq j \leq k-2$) need to be computed at iteration k as the other quantities are available from the previous iteration.

A separate calculation may be used to update the *diagonals* of B_k from (3.8). On completion of iteration k , these diagonals form the next positive-definite $B_{k+1}^{(0)}$. Then, at the k th iteration, we define the approximate Hessian

$$B_k = B_k^{(k)} = B_k^{(0)} + V_k V_k^T - U_k U_k^T,$$

where $U_k = (u_{k-\ell} \ u_{k-\ell+1} \ \cdots \ u_{k-1})$ and $V_k = (v_{k-\ell} \ v_{k-\ell+1} \ \cdots \ v_{k-1})$. It must be emphasized that B_k is not computed explicitly. Many sparse QP solvers access B_k by requesting products of the form $B_k u$. These are computed with work proportional to ℓ . For situations where the QP solver solves an explicit sparse system of the form (3.7), the solution may be found using the bordered matrix

$$\begin{pmatrix} \tilde{B}_k^{(0)} & \tilde{J}_k^T & \tilde{V}_k^T & \tilde{U}_k^T \\ \tilde{J}_k & & & \\ \tilde{V}_k & I & & \\ \tilde{U}_k & & -I & \end{pmatrix} \begin{pmatrix} p_k \\ -\hat{\pi}_k \\ r \\ s \end{pmatrix} = - \begin{pmatrix} (g_k + B_k \eta_k)_B \\ c_k + J_k \eta_k \\ 0 \\ 0 \end{pmatrix},$$

where $\tilde{B}_k^{(0)}$, \tilde{J}_k , \tilde{V}_k and \tilde{U}_k denote the matrices of basic components of $B_k^{(0)}$, J_k , V_k and U_k . Following [88, Section 3.6.2], if we define

$$K_0 = \begin{pmatrix} \tilde{B}_k^{(0)} & \tilde{J}_k^T \\ \tilde{J}_k & \end{pmatrix}, \quad S = \begin{pmatrix} I & \\ & -I \end{pmatrix} - \begin{pmatrix} \tilde{V}_k^T \\ \tilde{U}_k^T \end{pmatrix} K_0^{-1} \begin{pmatrix} \tilde{V}_k & \tilde{U}_k \end{pmatrix},$$

it would be efficient to work with a sparse factorization of K_0 and dense factors of its Schur complement S . (For a given QP subproblem, U and V are constant, but changes to \tilde{J}_k would be handled by appropriate updates to the Schur complement. See Section A.4.2 of the Appendix.) For general QP solvers that require an explicit sparse Hessian, the limited-memory updates can be applied implicitly by including additional linear equality constraints in the QP subproblem, see Gould and Robinson [107]. Bradley [12] describes a BFGS limited-memory method for SQP that employs a diagonal approximation in conjunction with a circular buffer.

In practice, the quasi-Newton approximation may become indefinite because of rounding error and it is better numerically to write B_k in the form $B_k = G_k^T G_k$, where G_k is the product of elementary matrices

$$G_k = G_k^{(0)} \prod_{j=k-\ell}^{k-1} (I + d_j w_j^T), \quad (3.23)$$

with $B_k^{(0)} = G_k^{(0)T} G_k^{(0)}$ and $w_j = (\pm v_j - u_j)/(d_j^T B_k^{(j)} d_j)^{\frac{1}{2}}$ (see Brodlie, Gourlay and Greenstadt [13], Dennis and Schnabel [48], and Gill, Murray and Saunders [83]). The sign of v_j may be chosen to minimize the rounding error in computing w_j . The quantities (d_j, w_j) are stored for each j . During outer iteration k , the QP solver accesses B_k by requesting products of the form $B_k z$. These are computed with work proportional to ℓ using the recurrence relations:

$$\begin{aligned} z &\leftarrow z + (w_j^T z) d_j, \quad j = k-1:k-\ell; \quad z \leftarrow G_k^{(0)} z; \\ t &\leftarrow G_k^{(0)T} z; \quad t \leftarrow t + (d_j^T t) w_j, \quad j = k-\ell:k-1. \end{aligned}$$

Products of the form $u^T B_k u$ are easily and safely computed as $\|z\|_2^2$ with $z = G_k u$.

In a QP solver that updates the Schur complement matrix an explicit sparse Hessian, the system (3.7) with $B_k = G_k^T G_k$ is equivalent to

$$\left(\begin{array}{c|c} \tilde{B}_k^{(0)} & \tilde{J}_k^T \\ \hline \tilde{J}_k & \end{array} \right) \left(\begin{array}{ccccc} \tilde{u}_{k-\ell} & \tilde{w}_{k-\ell} & \cdots & \tilde{u}_{k-1} & \tilde{w}_{k-1} \end{array} \right) \left(\begin{array}{c} p_k \\ -\hat{\pi}_k \\ r_{k-\ell} \\ s_{k-\ell} \\ \vdots \\ r_{k-1} \\ s_{k-1} \end{array} \right) = - \left(\begin{array}{c} (g_k + B_k \eta_k)_B \\ c_k + J_k \eta_k \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{array} \right),$$

where $\tilde{d}_j = (d_j)_B$, $\tilde{u}_j = (B_k^{(j)} d_j)_B$, and $\gamma_j = \tilde{d}_j^T \tilde{u}_j$ (see Gill, Murray and Saunders [83], and Huynh [123]).

An alternative form of the limited-memory update is used by Gill, Murray and Saunders [83]. Let r and k denote two outer iterations such that $r \leq k \leq r + \ell$. At iteration k the BFGS approximate Hessian may be expressed in terms of ℓ updates to a positive-definite B_r :

$$B_k = B_r + \sum_{i=r}^{k-1} (v_i v_i^T - u_i u_i^T), \quad (3.24)$$

where $u_i = B_i d_i / (d_i^T B_i d_i)^{\frac{1}{2}}$, and $v_i = y_i / (y_i^T d_i)^{\frac{1}{2}}$. In this scheme, the $k - r$ pairs (u_i, v_i) do not need to be recomputed for each update. On completion of iteration $k = r + \ell$, a total of ℓ pairs have been accumulated, and the storage is “reset” by discarding the previous updates. Moreover, the definition of u_i is simplified by the identity $B_i d_i = -\alpha_i \nabla_x L(\hat{x}_i, \hat{\pi}_i, \hat{z}_i)$ that follows from the QP optimality conditions (2.18). As in the previous scheme, a separate calculation may be used to update the diagonals of B_k from (3.8). On completion of iteration $k = r + \ell$, these diagonals form the next positive-definite B_r (with $r = k + 1$).

This scheme has an advantage in the SQP context when the constraints are linear: the reduced Hessian for the QP subproblem can be updated between outer iterations (see Section A.4.1).

Early termination of QP subproblems. SQP theory usually assumes that the QP subproblems are solved to optimality. For large problems with a poor starting point and $B_0 = I$, many thousands of iterations may be needed for the first QP, building up many free variables that are promptly eliminated by more thousands of iterations in the second QP. In general, it seems wasteful to expend much effort on any QP before updating B_k and the constraint linearization.

Any scheme for early termination must be implemented in a way that does not compromise the reliability of the SQP method. For example, suppose that the QP iterations are terminated after an arbitrary fixed number of steps. If a primal active-set method is used to solve the subproblem, the multipliers associated with QP constraints that have not been optimized will be negative. Using these multipliers directly (or first setting them to zero) in the definition of the Lagrangian function is problematic. The resulting search direction may not be a descent direction for the merit function, or may require the penalty parameter to be increased unnecessarily. For example, the value of the lower bound on the penalty parameter for the ℓ_1 merit function involves the values of the QP multipliers—see, (3.13). Dembo and Tulowitzki [47] suggest using a dual feasible active-set method for the QP subproblem and terminating the inner iterations when the norm of a potential search direction $p_k = \hat{x}_k - x_k$ is small. Dual feasible active-set methods have the advantage that the approximate multipliers are nonnegative, but a terminated iteration will have some negative primal variables—this time making the definition of the search direction problematic.

Murray and Prieto [142] suggest another approach to terminating the QP solutions early, requiring that at least one QP subspace stationary point be reached (see Definition A.1 of the Appendix). The associated theory implies that any subsequent point \hat{x}_k generated by a special-purpose primal-feasible QP solver gives a sufficient decrease in the augmented Lagrangian merit function (3.16), provided that $\|\hat{x}_k - x_k\|$ is nonzero.

Another way to save inner iterations safely during the early outer iterations is to *suboptimize* the QP subproblem. At the start of an outer iteration, many variables are fixed at their current values (i.e., x_i is fixed at $(x_k)_i$) and an SQP outer iteration is performed on the reduced problem (solving a smaller QP to get a search direction for the nonfixed variables). Once a solution of the reduced QP is found, the fixed variables are freed, and the outer iteration is completed with a “full” search direction that happens to leave many variables unaltered because $p_i = (\hat{x}_i - x_k)_i = 0$ for the temporarily fixed variables. At each step, the conventional theory for the reduction in the merit function should guarantee progress on the associated reduced *nonlinear* problem. In practice, it may not be obvious which variables should be fixed at each stage, the reduced QP could be infeasible, and degeneracy could produce a zero search direction. Instead, the choice of which variables to fix is made within the QP solver. In the method of SNOPT, QP iterations are performed on the full problem until a feasible point is found or elastic mode is entered. The iterations are continued until certain limits are reached and not all steps have been degenerate. At this point all variables such that $x_i = (x_k)_i$ are frozen at their current value and the reduced QP is solved to optimality. With this scheme it is safe to impose rather arbitrary limits, such as limits on the number of iterations (for the various termination conditions that may be applied, see Gill, Murray and Saunders [83, 84]). Note that this form of suboptimization enforces the condition $((\hat{x}_k - x_k) \cdot \hat{z}_k)_i = 0$ for the frozen variables and so the nonoptimized variables have no affect on the magnitude of the penalty parameter in (3.13).

3.3. Sequential unconstrained methods. Fletcher [61] observed that the ℓ_1 penalty function (1.8) can be minimized subject to bounds by solving a sequence of nondifferentiable *unconstrained* subproblems of the form

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \widehat{f}_k(x) + \rho \|\widehat{c}_k(x)\|_1 + \rho \| [x]_- \|_1, \quad (3.25)$$

where $\widehat{c}_k(x)$ denotes the linearized constraint functions $\widehat{c}_k(x) = c_k + J_k(x - x_k)$, and $[v]_- = \max\{-v_i, 0\}$. In this case the bound constraints are *not* imposed explicitly. Fletcher proposed minimizing this function using a trust-region method, although a line-search method would also be appropriate, particularly if H_k were positive definite. The trust-region subproblem has the form

$$\begin{aligned} & \underset{d \in \mathbb{R}^n}{\text{minimize}} \quad \hat{f}_k(x_k + d) + \rho \|\hat{c}_k(x_k + d)\|_1 + \rho \| [x_k + d]_- \|_1 \\ & \text{subject to} \quad \|d\| \leq \delta_k. \end{aligned} \quad (3.26)$$

The trust-region radius δ_k is increased or decreased based on condition (3.5), where $\mathcal{M}(x; \rho) = P_1(x; \rho)$ and $m_k(x; \rho)$ is defined in terms of an affine or quadratic model of the modified Lagrangian (see (3.12)). This approach forms the basis of Fletcher's “ $S\ell_1$ -QP method”. Each subproblem has a piecewise quadratic objective function and has the same complexity as a quadratic program of the form (2.1). If the infinity norm is used to define the size of the trust region, the subproblem is equivalent to the smooth quadratic program

$$\begin{aligned} & \underset{d, w \in \mathbb{R}^n; u, v \in \mathbb{R}^m}{\text{minimize}} \quad \hat{f}_k(x_k + d) + \rho e^T u + \rho e^T v + \rho e^T w \\ & \text{subject to} \quad \hat{c}_k(x_k + d) - u + v = 0, \quad u \geq 0, \quad v \geq 0, \\ & \quad -\delta_k e \leq d \leq \delta_k e, \quad x_k + d + w \geq 0, \quad w \geq 0, \end{aligned}$$

where, analogous to (1.4), the vectors u and v may be interpreted as the positive and negative parts of the affine function $c_k + J_k d$. A benefit of this formulation is that a solution of (3.26) always exists, even when the linearized constraints of the plain QP subproblem are inconsistent. Observe that the unconstrained subproblem (3.25) is defined in terms of $\hat{f}_k(x)$, the model of the modified Lagrangian (see (2.2)). This feature is crucial because it implies that if the trust-region radius and penalty parameter are sufficiently large in the neighborhood of an isolated solution, the $S\ell_1$ -QP subproblem is the same as the plain SQP subproblem (2.1). Nevertheless, the implicit minimization of the ℓ_1 penalty function means that there is the possibility of the Maratos effect. For the trust-region approach, the second-order correction may be determined from the quadratic program

$$\begin{aligned} & \underset{s, w \in \mathbb{R}^n; u, v \in \mathbb{R}^m}{\text{minimize}} \quad \hat{f}_k(x_k + d_k + s) + \rho e^T u + \rho e^T v + \rho e^T w \\ & \text{subject to} \quad J_k s - u + v = -c(x_k + d_k), \quad u \geq 0, \quad v \geq 0, \\ & \quad -\delta_k e \leq d_k + s \leq \delta_k e, \quad x_k + d_k + s \geq 0, \quad w \geq 0. \end{aligned} \quad (3.27)$$

Yuan [177] gives an analysis of the superlinear convergence of trust-region methods that use the second-order correction.

The $S\ell_1$ -QP approach can be used in conjunction with other unconstrained merit functions. Many of these extensions lead to a subproblem that is equivalent to a quadratic program. The “ $S\ell_\infty$ -QP method” uses the trust-region subproblem

$$\begin{aligned} & \underset{d \in \mathbb{R}^n}{\text{minimize}} \quad \hat{f}_k(x_k + d) + \rho \|\hat{c}_k(x_k + d)\|_\infty + \rho \| [x_k + d]_- \|_\infty \\ & \text{subject to} \quad \|d\|_\infty \leq \delta_k, \end{aligned} \quad (3.28)$$

which is equivalent to the quadratic program

$$\begin{aligned} & \underset{d \in \mathbb{R}^n; \theta, \sigma \in \mathbb{R}}{\text{minimize}} \quad \hat{f}_k(x_k + d) + \rho\theta + \rho\sigma \\ & \text{subject to} \quad -\theta e \leq \hat{c}_k(x_k + d) \leq \theta e, \quad \theta \geq 0, \\ & \quad -\delta_k e \leq d \leq \delta_k e, \quad x_k + d + \sigma e \geq 0, \quad \sigma \geq 0, \end{aligned} \tag{3.29}$$

see, e.g., Yuan [178], and Exler and Schittkowski [57]. A QP problem for the second-order correction may be defined analogous to (3.27). For a general discussion of the convergence properties of nondifferentiable exact penalty functions in the SQP context, see Fletcher [64], Burke [22], and Yuan [176].

3.4. Filter methods. The definition of the merit function in the Han-Powell method or the nonsmooth objective function in the sequential unconstrained optimization method requires the specification of a penalty parameter that weights the effect of the constraint violations against the value of the objective. Another way of forcing convergence is to use a *filter*, which is a two-dimensional measure of quality based on $f(x)$ and $\|c(x)\|$, where we assume that $x \geq 0$ is satisfied throughout. A filter method requires that progress be made with respect to the two-dimensional function $(\|c(x)\|, f(x))$. Using the conventional notation of filter methods, we define $h(x) = \|c(x)\|$ as the measure of infeasibility of the equality constraints, and use (h_j, f_j) to denote the pair $(h(x_j), f(x_j))$.

The two-dimensional measure provides the conditions for a point \bar{x} to be “better” than a point \hat{x} . Given two points \bar{x} and \hat{x} , the pair $(h(\bar{x}), f(\bar{x}))$ is said to *dominate* the pair $(h(\hat{x}), f(\hat{x}))$ if

$$h(\bar{x}) \leq \beta h(\hat{x}) \quad \text{and} \quad f(\bar{x}) \leq f(\hat{x}) - \gamma h(\bar{x}),$$

where $\beta, \gamma \in (0, 1)$ are constants with $1 - \beta$ and γ small (e.g., $\beta = 1 - \gamma$ with $\gamma = 10^{-3}$). (For brevity, we say that \bar{x} dominates \hat{x} , although it must be emphasized that only the objective value and constraint norm are stored.) A filter \mathcal{F} consists of a list of entries (h_j, f_j) such that no entry dominates another. (This filter is the so-called *sloping filter* proposed by Chin [31] and Chin and Fletcher [32]. The original filter proposed by Fletcher and Leyffer [66, 67] uses $\gamma = 0$ and $\beta = 1$.)

A pair $(h(x_k), f(x_k))$ is said to be “acceptable to the filter” \mathcal{F} if and only if it is not dominated by any entry in the filter, i.e.,

$$h(x_k) \leq \beta h_j \quad \text{or} \quad f(x_k) \leq f_j - \gamma h(x_k) \tag{3.30}$$

for every $(h_j, f_j) \in \mathcal{F}$. In some situations, an accepted point $(h(x_k), f(x_k))$ is added to the filter. This operation adds $(h(x_k), f(x_k))$ to the list of entries (h_j, f_j) in \mathcal{F} , and removes any entries that are dominated by the new pair. The test (3.30) provides an important inclusion property that if a pair (h, f) is added to the filter, then the set of points that are unacceptable

for the new filter always includes the points that are unacceptable for the old filter.

As in the Burke-Han approach of Section 3.2.3, the principal goal of a filter method is the attainment of feasibility. An important property of the filter defined above is that if there are an infinite sequence of iterations in which $(h(x_k), f(x_k))$ is entered into the filter, and $\{f(x_k)\}$ is bounded below, then $h(x_k) \rightarrow 0$ (see Fletcher, Leyffer and Toint [68]).

3.4.1. Trust-region filter methods. Fletcher and Leyffer [66, 67] propose a trust-region filter method in which a filter is used to accept or reject points generated by a plain SQP subproblem with a trust-region constraint. Below we give a brief description of the variant of the Fletcher-Leyffer method proposed by Fletcher, Leyffer and Toint [68]. The filter is defined in terms of the one-norm of the constraint violations, i.e., $h(x) = \|c(x)\|_1$, and the trust-region subproblem is given by

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k(x - x_k) \\ & \text{subject to} \quad c_k + J_k(x - x_k) = 0, \quad x \geq 0, \quad \|(x - x_k)\|_\infty \leq \delta_k. \end{aligned} \quad (3.31)$$

To simplify the discussion, we start by assuming that the QP subproblem (3.31) remains feasible. In this case, the filter method generates a sequence of points $\{x_k\}$ and a corresponding sequence of filters $\{\mathcal{F}_k\}$ such that x_k is acceptable to the filter \mathcal{F}_k and $x_{k+1} = \hat{x}_k$, where \hat{x}_k is a global minimizer of the QP subproblem (3.31). The use of a filter alone does not necessarily enforce convergence to a solution of the constrained problem. For example, if the iterates converge to an arbitrary feasible point and the infeasibility measure h is reduced by a factor of at least β at each iteration, then the iterates will be acceptable to the filter independently of f . This implies that the filter must be used in conjunction with a sufficient reduction condition analogous to (3.1), i.e.,

$$\Delta m_k(d_k) > 0 \quad \text{and} \quad f(x_k) - f(x_k + d_k) \geq \eta \Delta m_k(d_k), \quad (3.32)$$

where $d_k = \hat{x}_k - x_k$, and $\Delta m_k(d_k) = m_k(x_k) - m_k(x_k + d_k)$ for a local model $m_k(x)$ of f (e.g., $m_k(x) = f(x_k) + g(x_k)^T(x - x_k)$).

At the start of the k th iteration, we have a point x_k and a filter \mathcal{F}_{k-1} such that (h_k, f_k) is acceptable to \mathcal{F}_{k-1} , but is not yet included in \mathcal{F}_{k-1} (it will be shown below that x_k may or may not be included in the filter even though it constitutes an acceptable entry). The k th iteration is analogous to that of a backtracking line-search method, except that the backtracking steps are performed by solving the QP (3.31) with decreasing values of the trust-region radius. The backtracking continues until \hat{x}_k is acceptable to the combined filter $\mathcal{F}_{k-1} \cup (h_k, f_k)$, and either $f(x_k) - f(x_k + d_k) \geq \eta \Delta m_k(d_k)$ or $\Delta m_k(d_k) \leq 0$. On termination of the backtracking procedure, if $\Delta m_k(d_k) \leq 0$, then (h_k, f_k) is added to \mathcal{F}_{k-1} (giving \mathcal{F}_k), otherwise $\mathcal{F}_k = \mathcal{F}_{k-1}$. Finally, the next iterate is defined

as $x_{k+1} = \hat{x}_k$ and the trust-region radius δ_{k+1} for the next iteration is initialized at some value greater than some preassigned minimum value δ_{\min} . This reinitialization provides the opportunity to increase the trust-region radius based on the change in f . For example, the trust-region radius can be increased if the predicted reduction in f is greater than some positive factor of h .

As mentioned above, although (h_k, f_k) is acceptable to \mathcal{F}_{k-1} , it is not necessarily added to the filter. The point x_k is added if and only if $\Delta m_k(d_k) \leq 0$, in which case the QP solution predicts an increase in f , and the primary aim of the iteration changes to that of reducing h (by allowing f to increase if necessary). The requirement that $\Delta m_k(d_k) \leq 0$ for adding to the filter ensures that all the filter entries have $h_j > 0$. This is because if $h_k = 0$, then the QP must be compatible (even without this being an assumption), and hence, if x_k is not a KKT point, then $\Delta m_k(d_k) > 0$ and x_k is not added to the filter.

Now we drop our assumption that the QP problem (3.31) is always feasible. If a new entry is never added to the filter during the backtracking procedure, then $\delta_k \rightarrow 0$ and there are two situations that can occur. If $c(x_k) = 0$, then the problem looks like an unconstrained problem. If f is reduced then we must make progress and conventional trust-region theory applies. On the other hand, if $c(x_k) \neq 0$, then reducing the trust-region radius will eventually give an infeasible QP. In this case, the method switches to a *restoration phase* that focuses on minimizing $h(x)$ subject to $x \geq 0$. In this case a *restoration filter* may be defined that allows nonmonotone progress on $h(x)$. Note that it is possible for the QP to be infeasible for any infeasible x_k . In this situation the filter method will converge to a nonoptimal local minimizer of $h(x)$ (just as the Han-Powell method may converge to a nonoptimal local minimizer of the merit function).

The convergence properties of filter-SQP methods are similar to those of methods that use a merit function. In particular, it is possible to establish convergence to either a point that satisfies the first-order necessary conditions for optimality, or a point that minimizes $h(x)$ locally (see Fletcher, Leyffer and Toint [68] for details). It is not necessary that H_k be positive definite, although \hat{x}_k must be a global solution of the QP subproblem (3.31) (see the cautionary opening remarks of the Appendix concerning the solution of indefinite QPs). Standard examples that exhibit the Maratos effect for an SQP method with a merit function cause no difficulties for the filter method. Although the unit step causes an increase in the constraint violation, and hence an increase in a penalty function, it also causes a decrease in the objective and so it is acceptable to the filter. However, Fletcher and Leyffer [67] give a simple example for which the QP solution increases both the objective and the constraint violations, resulting in a reduction in the trust-region radius and the rejection of the Newton step. Fletcher and Leyffer propose the use of a second-order correction step analogous to (3.27). Ulbrich [168] defines a filter that uses the Lagrangian function instead of

f and shows that superlinear convergence may be obtained without using the second-order correction.

3.4.2. Line-search filter methods. The trust-region filter method described in Section 3.4.1 may be modified to use a line search by solving the plain SQP subproblem and replacing the backtracking trust-region procedure by a conventional backtracking line search. In this case, the candidate pair for the filter is $(h(x_k + \alpha_k p_k), f(x_k + \alpha_k p_k))$, where α_k is a member of a decreasing sequence of steplengths, and $p_k = \hat{x}_k - x_k$, with \hat{x}_k a solution of the plain QP (2.17). Analogous to (3.32), the sufficient decrease criteria for the objective are

$$\Delta m_k(\alpha_k p_k) > 0 \quad \text{and} \quad f(x_k) - f(x_k + \alpha_k p_k) \geq \eta \Delta m_k(\alpha_k p_k).$$

If the trial step length is reduced below a minimum value α_k^{\min} , the line search is abandoned and the algorithm switches to the restoration phase. For more details, the reader is referred to the two papers of Wächter and Biegler [171, 170]. The caveats of the previous section concerning the definition of H_k also apply to the line-search filter method. In addition, the absence of an explicit bound on $\|x - x_k\|$ provided by the trust-region constraint adds the possibility of unboundedness of the QP subproblem.

Chin, Rashid and Nor [33] consider a line-search filter method that includes a second-order correction step during the backtracking procedure. If $x_k + \alpha p_k$ is not acceptable to the filter, a second-order correction s_k is defined by solving the equality-constrained QP:

$$\begin{aligned} & \underset{s \in \mathbb{R}^n}{\text{minimize}} \quad f_k + g_k^T(p_k + s) + \frac{1}{2}(p_k + s)^T H_k(p_k + s) \\ & \text{subject to} \quad c(x_k + p_k) + J_k s = 0, \quad (x_k + p_k + s)_A = -\|p_k\|^\nu e, \end{aligned} \tag{3.33}$$

where $\nu \in (2, 3)$ and $\mathcal{A}(\hat{x}_k)$ is the active set predicted by the QP subproblem (for a similar scheme, see Herskovits [121], and Panier and Tits [147, 148]). Given an optimal solution s_k , Chin, Rashid and Nor [33] show that under certain assumptions, the sufficient decrease criteria

$$f(x_k) - f(x_k + \alpha_k p_k + \alpha_k^2 s_k) \geq \eta \Delta m_k(\alpha_k p_k) \quad \text{and} \quad \Delta m_k(\alpha_k p_k) > 0$$

give a sequence $\{x_k\}$ with local Q-superlinear convergence.

3.5. SQP methods based on successive LP and QP. In the MINLP context, it is necessary to solve a sequence of related nonlinear programs, some with infeasible constraints. For maximum efficiency, it is crucial that the active set from one problem is used to provide a warm start for the next. A substantial benefit of SQP methods is that they are easily adapted to accept an estimate of the active set. However, if warm starts are to be exploited fully, it is necessary that the second derivatives of the problem functions are available and that these derivatives are utilized by the SQP method. Unfortunately, none of the SQP methods discussed

in Sections 3.2–3.4 are completely suitable for use with second derivatives. The main difficulty stems from the possibility that the Hessian of the Lagrangian is indefinite, in which case the inequality constrained QP subproblem is nonconvex. A nonconvex QP is likely to have many local solutions, and may be unbounded. Some SQP methods are only well-defined if the subproblem is convex—e.g., methods that rely on the use of a positive-definite quasi-Newton approximate Hessian. Other methods require the calculation of a global solution of the QP subproblem, which has the benefit of ensuring that the “same” local solution is found for the final sequence of related QPs. Unfortunately, nonconvex quadratic programming is NP-hard, and even the seemingly simple task of checking for local optimality is intractable when there are zero Lagrange multipliers (see the opening remarks of the Appendix).

One approach to resolving this difficulty is to estimate the active set using a *convex programming approximation* of the plain QP subproblem (2.1). This active set is then used to define an equality-constrained QP (EQP) subproblem whose solution may be used in conjunction with a merit function or filter to obtain the next iterate. One of the first methods to use a convex program to estimate the active set was proposed by Fletcher and Sainz de la Maza [69], who proposed estimating the active set by solving a linear program with a trust-region constraint. (Their method was formulated first as a sequential unconstrained method for minimizing a nonsmooth composite function. Here we describe the particular form of the method in terms of minimizing the ℓ_1 penalty function $P_1(x, \rho)$ defined in (1.8).) The convex subproblem has the form

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad l_k(x) = f_k + g_k^T(x - x_k) + \rho \|\hat{c}_k(x)\|_1 + \rho \| [x]_- \|_1 \\ & \text{subject to} \quad \|x - x_k\| \leq \delta_k, \end{aligned} \tag{3.34}$$

which involves the minimization of a piecewise linear function subject to a trust-region constraint (cf. (3.26)). If the trust-region constraint is defined in terms of the infinity-norm, the problem (3.34) is equivalent to the linear programming (LP) problem:

$$\begin{aligned} & \underset{x, w \in \mathbb{R}^n; u, v \in \mathbb{R}^m}{\text{minimize}} \quad f_k + g_k^T(x - x_k) + \rho e^T u + \rho e^T v + \rho e^T w \\ & \text{subject to} \quad c_k + J_k(x - x_k) - u + v = 0, \quad u \geq 0, \quad v \geq 0, \\ & \quad x_k - \delta_k e \leq x \leq x_k + \delta_k e, \quad x + w \geq 0, \quad w \geq 0. \end{aligned} \tag{3.35}$$

This equivalence was the motivation for the method to be called the *successive linear programming* (SLP) method. Fletcher and Sainz de la Maza use the reduction in $P_1(x, \rho)$ predicted by the *first-order* subproblem (3.34) to assess the quality of the reduction $P_1(x_k, \rho) - P_1(x_k + d_k, \rho)$ defined by a *second-order* method (to be defined below).

Given a positive-definite approximation B_k of the Hessian of the Lagrangian, let $q_k(x)$ denote the piecewise quadratic function

$$q_k(x) = l_k(x) + \frac{1}{2}(x - x_k)^T B_k (x - x_k).$$

Let $d_k^{\text{LP}} = \hat{x}_k^{\text{LP}} - x_k$, where \hat{x}_k^{LP} is a solution of the LP (3.35), and define $\Delta l_k = l_k(x_k) - l_k(x_k + d_k^{\text{LP}})$. Then it holds that

$$q_k(x_k) - \min_d q_k(x_k + d) \geq \frac{1}{2} \Delta l_k \min\{\Delta l_k / \beta_k, 1\},$$

where $\beta_k = (d_k^{\text{LP}})^T B_k d_k^{\text{LP}} > 0$. This inequality suggests that a suitable acceptance criterion for an estimate $x_k + d$ is

$$P_1(x_k, \rho) - P_1(x_k + d, \rho) \geq \eta \Delta l_k \min\{\Delta l_k / \beta_k, 1\},$$

where η is some preassigned scalar such that $0 < \eta < \frac{1}{2}$. This criterion is used to determine if the new iterate x_{k+1} should be set to (i) the current iterate x_k (which always triggers a reduction in the trust-region radius); (ii) the second-order step $x_k + d_k$; or (iii) the first-order step $x_k + d_k^{\text{LP}}$. The test for accepting the second-order step is done first, If the second-order step fails, then the penalty function is recomputed at $x_k + d_k^{\text{LP}}$ and the test is repeated to determine if x_{k+1} should be $x_k + d_k^{\text{LP}}$. Finally, the trust-region radius is updated based on a conventional trust-region strategy that compares the reduction in the penalty function with the reduction predicted by the LP subproblem (the reader is referred to the original paper for details).

Next, we consider how to define a second-order step. Let \mathcal{B} and \mathcal{N} denote the final LP basic and nonbasic sets for the LP (3.35). To simplify the description, assume that the optimal u , v and w are zero. A second-order iterate \hat{x}_k can be defined as the solution of an equality-constrained quadratic program (EQP) defined by minimizing the quadratic model $\hat{f}_k(x) = f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k)$ subject to $c_k + J_k(x - x_k) = 0$, with the nonbasic variables fixed at their current values. Let $p_k = \hat{x}_k - x_k$, where $(\hat{x}_k, \hat{\pi}_k)$ is the primal-dual EQP solution. Let \tilde{p}_k denote the vector of components of p_k in the final LP basic set \mathcal{B} , with \tilde{J}_k the corresponding columns of J_k . The vector $(\tilde{p}_k, \hat{\pi}_k)$ satisfies the KKT equations

$$\begin{pmatrix} \tilde{B}_k & \tilde{J}_k^T \\ \tilde{J}_k & 0 \end{pmatrix} \begin{pmatrix} \tilde{p}_k \\ -\hat{\pi}_k \end{pmatrix} = - \begin{pmatrix} (g_k + B_k \eta_k)_{\mathcal{B}} \\ c_k + J_k \eta_k \end{pmatrix}, \quad (3.36)$$

where η_k is defined in terms of the final LP nonbasic set, i.e.,

$$(\eta_k)_i = \begin{cases} (\hat{x}_k^{\text{LP}} - x_k)_i & \text{if } i \in \mathcal{N}; \\ 0 & \text{if } i \notin \mathcal{N}. \end{cases}$$

There are many ways of solving these KKT equations. The most appropriate method will depend on certain basic properties of the problem being

solved, which include the size of the problem (i.e., the number of variables and constraints); whether or not the Jacobian is dense or sparse; and how the approximate Hessian is stored (see, e.g., Section 3.2.1). Fletcher and Sainz de la Maza suggest finding an approximate solution of the EQP using a quasi-Newton approximation of the reduced Hessian matrix (see Coleman and Conn [36]).

The results of Fletcher and Sainz de la Maza may be used to show that, under reasonable nondegeneracy and second-order conditions, the active set of the LP subproblem (3.35) ultimately predicts that of the smooth variant of the penalty function at limit points of $\{x_k\}$. This implies fast asymptotic convergence. Fletcher and Sainz de la Maza did not consider the use of exact second derivatives in their original paper, and it took more than 12 years before the advent of reliable second-derivative trust-region and filter methods for the EQP subproblem allowed the potential of SLP methods to be realized. Chin and Fletcher [32] proposed the use of a trust-region filter method that does not require the use of the ℓ_1 penalty function. For a similar approach that uses a filter, see Fletcher et al. [65]. In a series of papers, Byrd, Gould, Nocedal and Waltz [25, 26] proposed a method that employs an additional trust region to safeguard the EQP direction. They also define an appropriate method for adjusting the penalty parameter. Recently, Morales, Nocedal and Wu [137], and Gould and Robinson [105, 106, 107] have proposed SQP methods that identify the active set using a convex QP based on a positive-definite BFGS approximation of the Hessian.

4. SQP issues relevant to MINLP.

4.1. Treatment of linear constraints. An important feature of SQP methods is that it is relatively easy to exploit the special properties of linear constraints. This can be an advantage when a method for MINLP solves a sequence of NLPs that differ by only the number of linear constraints that are imposed. Suppose that the general linear constraints are a subset of the constraints defined by $c(x) = 0$, e.g., $c_L(x) = Ax - b = 0$. Then a feasible point for the linear constraints $c_L(x) = Ax - b = 0$, $x \geq 0$, can be found by solving the elastic problem

$$\begin{aligned} & \underset{x \in \mathbb{R}^n; u, v \in \mathbb{R}^m}{\text{minimize}} && \rho e^T u + e^T v \\ & \text{subject to} && Ax - u + v = b, \quad x \geq 0, \quad u \geq 0, \quad v \geq 0. \end{aligned} \tag{4.1}$$

This is equivalent to minimizing the one-norm of the general linear constraint violations subject to the simple bounds. An important property of linear constraints is that it is possible to determine the solvability of a system of linear inequalities in a finite number of steps. If the linear constraints are infeasible ($u \neq 0$ or $v \neq 0$), then the SQP algorithm can terminate without computing the nonlinear functions. Otherwise, all subsequent iterates satisfy the linear constraints.

4.2. Treatment of infeasibilities. If the constraints of the QP subproblem (2.1) have no feasible point, then no QP solution exists. This could be for two reasons, either: (i) the NLP is feasible but the quadratic programming subproblem is locally infeasible, or (ii) the NLP is infeasible. If the NLP is convex, then infeasibility of the quadratic programming subproblem implies infeasibility of the original problem, but in the nonconvex case, there is no such implication.

If the subproblem is infeasible, the algorithm may continue in *elastic mode*, by solving the elastic QP (3.22). There are two interpretations of the role of the elastic QP. In one interpretation, the elastic problem defines a regularization of the plain QP subproblem (2.1). In this case, if the NLP is feasible and $\rho_k \geq \|\pi_{k+1}\|_\infty$, then problems (2.1) and (3.22) are equivalent. An alternative interpretation is to view the elastic QP as the QP subproblem associated with the elastic *nonlinear* problem (1.4), so that the elastic constraints are present in the original problem and are inherited by the QP subproblem. Note that any solution of the NLP may be regarded as a solution of (3.22) for a value of ρ_k such that $\rho_k \geq \|\pi_{k+1}\|_\infty$. Hence, even if ρ_k is not present explicitly, we may consider both the subproblem (3.22) and the original problem (1.4) to have an associated implicit value of ρ_k that is larger than $\|\pi_{k+1}\|_\infty$.

4.3. Infeasibility detection. As we discussed in Section 1.3, it is important to be able to determine as quickly as possible if the NLP (1.2) is infeasible. In an SQP framework, infeasibility may be detected either by solving the QP subproblem in elastic mode (3.22) with a sequence of penalty parameters $\rho_k \rightarrow \infty$, or by solving a sequence of elastic *nonlinear* problems of the form (1.4) with $\rho_k \rightarrow \infty$. For an SQP method that solves a sequence of nonlinear elastic problems and uses a *quasi-Newton approximation* to the Hessian, infeasibility is usually signaled by the occurrence of a sequence of elastic problems in which the penalty parameter is increased, but the current x_k remains fixed, i.e., an optimal solution for a problem with $\rho = \rho_k$ is optimal for the problem with $\rho = \rho_{k+1}$, etc. This is usually a reliable indication that x_k is a local minimizer of the sum of infeasibilities. This behavior can be explained by the fact that a warm start uses the approximate Hessian from the previous elastic problem, which is not changed as ρ_k and the QP-multipliers are increased. This is one situation where the inability of a quasi-Newton Hessian to adapt to changes in the multipliers is beneficial!

The situation is different when the SQP method uses the exact Hessian of the Lagrangian. In this case, the multipliers reflect the magnitude of ρ_k , and so the Hessian changes substantially. In the following, we give a brief discussion of this case that reflects the paper of Byrd, Curtis and Nocedal [24]. For an infeasible problem, it must hold that $\rho_k \rightarrow \infty$ and $\rho_k > \rho_{k-1}$ for an infinite subsequence of iterates. In this situation, *different* problems are being solved at outer iterations $k-1$ and k . At iteration $k-1$,

the problem is the elastic problem (1.4) with $\rho = \rho_{k-1}$, whereas at iteration k , the problem is the elastic problem with $\rho = \rho_k$. We may write

$$f(x) + \rho_k e^T u + \rho_k e^T v = f(x) + \frac{\rho_k}{\rho_{k-1}} (\rho_{k-1} e^T u + \rho_{k-1} e^T v). \quad (4.2)$$

If the NLP is infeasible, it must hold that $\|u\| + \|v\| > 0$. If ρ_{k-1} is large, with $\rho_k > \rho_{k-1}$ and $\|u\| + \|v\| > 0$, then the term $f(x)$ is negligible in (4.2), i.e., $f(x) \ll \rho_{k-1} e^T u + \rho_{k-1} e^T v$, so that

$$\begin{aligned} f(x) + \rho_k e^T u + \rho_k e^T v &\approx \rho_k e^T u + \rho_k e^T v \\ &= \frac{\rho_k}{\rho_{k-1}} (\rho_{k-1} e^T u + \rho_{k-1} e^T v) \\ &\approx \frac{\rho_k}{\rho_{k-1}} (f(x) + \rho_{k-1} e^T u + \rho_{k-1} e^T v). \end{aligned} \quad (4.3)$$

The form of (4.4) implies that the elastic problems at iterations $k-1$ and k differ (approximately) by only a multiplicative factor ρ_k/ρ_{k-1} in the scaling of the objective function. The approximation becomes increasingly accurate as ρ_{k-1} tends to infinity.

Let (x_k, u_k, v_k) be the solution provided by the elastic QP subproblem at iteration $k-1$, with corresponding Lagrange multiplier estimates (π_k, z_k) . Also assume that (x_k, u_k, v_k) is close to optimal for the corresponding elastic problem (1.4) with $\rho = \rho_{k-1}$. If $\rho_k > \rho_{k-1}$, the question is how to provide a good initial point to this new problem. If (x_k, u_k, v_k) is the exact solution of the elastic problem for $\rho = \rho_{k-1}$, then (π_k, z_k) are the corresponding Lagrange multipliers. Moreover, if the objective functions differ by the factor ρ_k/ρ_{k-1} , then (x_k, u_k, v_k) is again optimal for the new problem, and the dual variables inherit the same scaling as the objective function (see (1.6b)). In this situation, the new multipliers are $((\rho_k/\rho_{k-1})\pi_k, (\rho_k/\rho_{k-1})z_k)$. Based on these observations, in an idealized situation, we expect that (x_k, u_k, v_k) , together with scaled Lagrange multiplier estimates $(\rho_k/\rho_{k-1})\pi_k$ and $(\rho_k/\rho_{k-1})z_k$, provide good initial estimates for the new elastic QP subproblem. Hence, if second derivatives are used in the QP subproblem, the Hessian of the Lagrangian should be evaluated at (x_k, u_k, v_k) with Lagrange multiplier estimates $((\rho_k/\rho_{k-1})\pi_k, (\rho_k/\rho_{k-1})z_k)$ in order to obtain fast convergence as ρ_k increases.

As ρ_k tends to infinity, the objective function becomes less important compared to the penalty term in the objective of (1.4). Eventually only the infeasibilities matter, and the iterates converge to a local minimizer of the sum of infeasibilities. See Byrd, Curtis and Nocedal [24] for a detailed discussion on infeasibility detection, including a discussion on how to let $\rho_k \rightarrow \infty$ rapidly.

4.4. Solving a sequence of related QP subproblems. In MINLP branch and bound methods it is necessary to solve a sequence of NLPs that differ by a single constraint (see, e.g., Leyffer [128], and Goux and

Leyffer [111]). For example, at the solution of a relaxed problem, some integer variables take a non-integer value. The MINLP algorithm selects one of the integer variables that takes a non-integer value, say x_i with value \bar{x}_i , and branches on it. Branching generates two new NLP problems by adding simple bounds $x_i \leq \lfloor \bar{x}_i \rfloor$ and $x_i \geq \lfloor \bar{x}_i \rfloor + 1$ to the NLP relaxation (where $\lfloor v \rfloor$ is the largest integer not greater than v). The SQP methods of Section 3.5 that solve an initial convex programming problem to determine the active set have the advantage that a dual QP/LP solver may be used to solve the convex QP subproblem (dual active-set QP methods are discussed in Section A.2). This provides similar advantages to MINLP solvers as the dual simplex method provides to MILP. If the SQP method is implemented with a dual QP solver, and is warm started with the primal-dual solution of the previous relaxation, then the dual variables are feasible and only one branched variable is infeasible. The infeasible x_i can be moved towards feasibility immediately.

A similar situation applies if a nonlinear cut adds a constraint to the NLP. For simplicity, assume that the QP has objective $g^T x + \frac{1}{2} x^T H x$ and constraints $Ax = b$, $x \geq 0$. As the QP is in standard form, the cut adds a new row and column to A , a zero element to the objective g , and a zero row and column to H . This gives a new problem with \bar{A} , \bar{b} , \bar{g} and \bar{H} (say). The new column of \bar{A} corresponds to the unit vector associated with the new slack variable. An obvious initial basis for the new problem is

$$\bar{A}_B = \begin{pmatrix} A_B & 0 \\ a^T & 1 \end{pmatrix},$$

so the new basic solution \bar{x}_B is the old solution x_B , augmented by the new slack, which is infeasible. This means that if we solve the primal QP then it would be necessary to go into phase 1 to get started. However, by solving the dual QP, then we have an initial feasible subspace minimizer for the dual based on a $\bar{y}_B (= \bar{x}_B)$ such that $\bar{A}_B \bar{y}_B = \bar{b}$ and

$$\bar{z} = \bar{g} + \bar{H} \bar{y} - \bar{A}^T \bar{\pi}.$$

We can choose $\bar{\pi}$ to be the old π augmented by a zero. The new element of \bar{y}_B corresponds to the new slack, so the new elements of \bar{g} and row and column of \bar{H} are zero. This implies that \bar{z} is essentially z , and hence $\bar{z} \geq 0$.

Acknowledgments. The authors would like to thank Anders Forsgren for numerous discussions on SQP methods during the preparation of this paper. We are also grateful to the referees for suggestions that substantially improved the presentation.

APPENDIX

A. Methods for quadratic programming. We consider methods for the quadratic program

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad g^T(x - x_I) + \frac{1}{2}(x - x_I)^T H(x - x_I) \\ & \text{subject to} \quad Ax = Ax_I - b, \quad x \geq 0, \end{aligned} \tag{A.1}$$

where g , H , b , A and x_I are given constant quantities, with H symmetric. The QP objective is denoted by $\hat{f}(x)$, with gradient $\hat{g}(x) = g + H(x - x_I)$. In some situations, the general constraints will be written as $\hat{c}(x) = 0$, with $\hat{c}(x) = A(x - x_I) + b$. The QP active set is denoted by $\mathcal{A}(x)$. A primal-dual QP solution is denoted by (x^*, π^*, z^*) . In terms of the QP defined at the k th outer iteration of an SQP method, we have $x_I = x_k$, $b = c(x_k)$, $g = g(x_k)$, $A = J(x_k)$ and $H = H(x_k, \pi_k)$. It is assumed that A has rank m . No assumptions are made about H other than symmetry. Conditions that must hold at an optimal solution of (A.1) are provided by the following result (see, e.g., Borwein [11], Contesse [43] and Majthay [132]).

RESULT A.1 (QP optimality conditions). *The point x^* is a local minimizer of the quadratic program (A.1) if and only if*

- (a) $\hat{c}(x^*) = 0$, $x^* \geq 0$, and there exists at least one pair of vectors π^* and z^* such that $\hat{g}(x^*) - A^T \pi^* - z^* = 0$, with $z^* \geq 0$, and $z^* \cdot x^* = 0$;
- (b) $p^T H p \geq 0$ for all nonzero p satisfying $\hat{g}(x^*)^T p = 0$, $A p = 0$, and $p_i \geq 0$ for every $i \in \mathcal{A}(x^*)$. \square

Part (a) gives the first-order KKT conditions (2.18) for the QP (A.1). If H is positive semidefinite, the first-order KKT conditions are both necessary and sufficient for (x^*, π^*, z^*) to be a local primal-dual solution of (A.1).

Suppose that (x^*, π^*, z^*) satisfies condition (a) with $z_i^* = 0$ and $x_i^* = 0$ for some i . If H is positive semidefinite, then x^* is a *weak minimizer* of (A.1). In this case, x^* is a global minimizer with a unique global minimum $\hat{f}(x^*)$. If H has at least one negative eigenvalue, then x^* is known as a *dead point*. Verifying condition (b) at a dead point requires finding the global minimizer of an indefinite quadratic form over a cone, which is an NP-hard problem (see, e.g., Cottle, Habetler and Lemke [44], Pardalos and Schnitger [149], and Pardalos and Vavasis [150]). This implies that the optimality of a candidate solution of a general quadratic program can be verified only if more restrictive (but computationally tractable) sufficient conditions are satisfied. A dead point is a point at which the sufficient conditions are not satisfied, but certain necessary conditions hold. Computationally tractable necessary conditions are based on the following result.

RESULT A.2 (Necessary conditions for optimality). *The point x^* is a local minimizer of the QP (A.1) only if*

- (a) $\hat{c}(x^*) = 0$, $x^* \geq 0$, and there exists at least one pair of vectors π^* and z^* such that $\hat{g}(x^*) - A^T \pi^* - z^* = 0$, with $z^* \geq 0$, and $z^* \cdot x^* = 0$;

- (b) $p^T H p \geq 0$ for all nonzero p satisfying $Ap = 0$, and $p_i = 0$ for every $i \in \mathcal{A}(x^*)$. \square

Suitable sufficient conditions for optimality are given by (a)–(b) with (b) replaced by the condition that $p^T H p \geq \omega \|p\|^2$ for some $\omega > 0$ and all p such that $Ap = 0$, and $p_i = 0$ for every $i \in \mathcal{A}_+(x)$, where $\mathcal{A}_+(x)$ is the index set $\mathcal{A}_+(x) = \{i \in \mathcal{A}(x) : z_i > 0\}$.

Typically, software for general quadratic programming is designed to terminate at a dead point. Nevertheless, it is possible to define procedures that check for optimality at a dead point, but the chance of success in a reasonable amount of computation time depends on the dimension of the problem (see Forsgren, Gill and Murray [72]).

A.1. Primal active-set methods. We start by reviewing the properties of *primal-feasible active-set methods* for quadratic programming. An important feature of these methods is that once a feasible iterate is found, all subsequent iterates are feasible. The methods have two phases. In the first phase (called the *feasibility phase* or *phase one*), a feasible point is found by minimizing the sum of infeasibilities. In the second phase (the *optimality phase* or *phase two*), the quadratic objective function is minimized while feasibility is maintained. Each phase generates a sequence of inner iterates $\{x_j\}$ such that $x_j \geq 0$. The new iterate x_{j+1} is defined as $x_{j+1} = x_j + \alpha_j p_j$, where the *step length* α_j is a nonnegative scalar, and p_j is the *QP search direction*. For efficiency, it is beneficial if the computations in both phases are performed by the same underlying method. The two-phase nature of the algorithm is reflected by changing the function being minimized from a function that reflects the degree of infeasibility to the quadratic objective function. For this reason, it is helpful to consider methods for the optimality phase first.

At the j th step of the optimality phase, $\widehat{c}(x_j) = A(x_j - x_I) + b = 0$ and $x_j \geq 0$. The vector p_j is chosen to satisfy certain properties with respect to the objective and constraints. First, p_j must be a *direction of decrease* for \widehat{f} at x_j , i.e., there must exist a positive $\bar{\alpha}$ such that

$$\widehat{f}(x_j + \alpha p_j) < \widehat{f}(x_j) \quad \text{for all } \alpha \in (0, \bar{\alpha}].$$

In addition, $x_j + p_j$ must be feasible with respect to the general constraints, and feasible with respect to the bounds associated with a certain “working set” of variables that serves as an estimate of the optimal active set of the QP. Using the terminology of linear programming, we call this working set of variables the *nonbasic set*, denoted by $\mathcal{N} = \{\nu_1, \nu_2, \dots, \nu_{n_N}\}$. Similarly, we define the set \mathcal{B} of indices that are not in \mathcal{N} as the *basic set*, with $\mathcal{B} = \{\beta_1, \beta_2, \dots, \beta_{n_B}\}$, where $n_B = n - n_N$. Although \mathcal{B} and \mathcal{N} are strictly index sets, we will follow common practice and refer to variables x_{β_r} and x_{ν_s} as being “in \mathcal{B} ” and “in \mathcal{N} ” respectively.

With these definitions, we define the columns of A indexed by \mathcal{N} and \mathcal{B} , the *nonbasic* and *basic* columns of A , as $A_{\mathcal{N}}$ and $A_{\mathcal{B}}$, respectively. We re-

frain from referring to the nonbasic and basic sets as the “fixed” and “free” variables because some active-set methods allow some nonbasic variables to move (the simplex method for linear programming being one prominent example). An important attribute of the nonbasic set is that A_B has rank m , i.e., the rows of A_B are linearly independent. This implies that the cardinality of the nonbasic set must satisfy $0 \leq n_N \leq n - m$. It must be emphasized that our definition of \mathcal{N} does not require a nonbasic variable to be active (i.e., at its lower bound). Also, whereas the active set is defined uniquely at each point, there are many choices for \mathcal{N} (including the empty set). Given any n -vector y , the vector of *basic components* of y , denoted by y_B , is the n_B -vector whose j th component is component β_j of y . Similarly, y_N , the vector *nonbasic components* of y , is the n_N -vector whose j th component is component ν_j of y .

Given a basic-nonbasic partition of the variables, we introduce the definitions of stationarity and optimality with respect to a basic set.

DEFINITION A.1 (Subspace stationary point). *Let \mathcal{B} be a basic set defined at an \hat{x} such that $\tilde{c}(\hat{x}) = 0$. Then \hat{x} is a subspace stationary point with respect to \mathcal{B} (or, equivalently, with respect to A_B) if there exists a vector π such that $\hat{g}_B(\hat{x}) = A_B^T \pi$. Equivalently, \hat{x} is a subspace stationary point with respect to \mathcal{B} if the reduced gradient $Z_B^T \hat{g}_B(\hat{x})$ is zero, where the columns of Z_B form a basis for the null-space of A_B .* \square

If \hat{x} is a subspace stationary point, \hat{f} is stationary on the subspace $\{x : A(x - \hat{x}) = 0, x_N = \hat{x}_N\}$. At a subspace stationary point, it holds that $g(\hat{x}) = A^T \pi + z$, where $z_i = 0$ for $i \in \mathcal{B}$ —i.e., $z_B = 0$. Subspace stationary points may be classified based on the curvature of \hat{f} on the nonbasic set.

DEFINITION A.2 (Subspace minimizer). *Let \hat{x} be a subspace stationary point with respect to \mathcal{B} . Let the columns of Z_B form a basis for the null-space of A_B . Then \hat{x} is a subspace minimizer with respect to \mathcal{B} if the reduced Hessian $Z_B^T H Z_B$ is positive definite.* \square

If the nonbasic variables are active at \hat{x} , then \hat{x} is called a *standard* subspace minimizer. At a standard subspace minimizer, if $z_N \geq 0$ then \hat{x} satisfies the necessary conditions for optimality. Otherwise, there exists an index $\nu_s \in \mathcal{N}$ such that $z_{\nu_s} < 0$. If some nonbasic variables are not active at \hat{x} , then \hat{x} is called a *nonstandard* subspace minimizer.

It is convenient sometimes to be able to characterize the curvature of \hat{f} in a form that does not require the matrix Z_B explicitly. The *inertia* of a symmetric matrix X , denoted by $\text{In}(X)$, is the integer triple (i_+, i_-, i_0) , where i_+ , i_- and i_0 denote the number of positive, negative and zero eigenvalues of X . Gould [101] shows that if A_B has rank m and $A_B Z_B = 0$, then $Z_B^T H_B Z_B$ is positive definite if and only if

$$\text{In}(K_B) = (n_B, m, 0), \quad \text{where } K_B = \begin{pmatrix} H_B & A_B^T \\ A_B & 0 \end{pmatrix} \quad (\text{A.2})$$

(see Forsgren [70] for a more general discussion, including the case where A_B does not have rank m). Many algorithms for solving symmetric equations

that compute an explicit matrix factorization of K_B also provide the inertia as a by-product of the calculation, see, e.g., Bunch [17], and Bunch and Kaufman [18].

Below, we discuss two alternative formulations of an active-set method. Each generates a feasible sequence $\{x_j\}$ such that $x_{j+1} = x_j + \alpha_j p_j$ with $\hat{f}(x_{j+1}) \leq \hat{f}(x_j)$. Neither method requires the QP to be convex, i.e., H need not be positive semidefinite. The direction p_j is defined as the solution of an QP subproblem with equality constraints. Broadly speaking, the nonbasic components of p_j are *specified* and the basic components of p_j are adjusted to satisfy the general constraints $A(x_j + p_j) = Ax_I - b$. If p_B and p_N denote the basic and nonbasic components of p_j , then the nonbasic components are fixed by enforcing constraints of the form $p_N = d_N$, where d_N is a constant vector that characterizes the active-set method being used. The restrictions on p_j define constraints $Ap = 0$ and $p_N = d_N$. Any remaining degrees of freedom are used to define p_j as the direction that produces the largest reduction in \hat{f} . This gives the equality constrained QP subproblem

$$\underset{p}{\text{minimize}} \quad \hat{g}(x_j)^T p + \frac{1}{2} p^T H p \quad \text{subject to} \quad Ap = 0, \quad p_N = d_N.$$

In the following sections we define two methods based on alternative definitions of d_N . Both methods exploit the properties of a subspace minimizer (see Definition A.2) in order to simplify the linear systems that must be solved.

A.1.1. Nonbinding-direction methods. We start with a method that defines a change in the basic-nonbasic partition at every iteration. In particular, one of three changes occurs: (i) a variable is moved from the basic set to the nonbasic set; (ii) a variable is moved from the nonbasic set to the basic set; or (iii) a variable in the basic set is swapped with a variable in the nonbasic set. These changes result in a column being added, deleted or swapped in the matrix A_B .

In order to simplify the notation, we drop the subscript j and consider the definition of a single iteration that starts at the primal-dual point (x, π) and defines a new iterate $(\bar{x}, \bar{\pi})$ such that $\bar{x} = x + \alpha p$ and $\bar{\pi} = \pi + \alpha q_\pi$. A crucial assumption about (x, π) is that it is a subspace minimizer with respect to the basis \mathcal{B} . It will be shown that this assumption guarantees that the next iterate $(\bar{x}, \bar{\pi})$ (and hence each subsequent iterate) is also a subspace minimizer.

Suppose that the reduced cost associated with the s th variable is negative, i.e., $z_{\nu_s} < 0$. The direction p is defined so that all the nonbasic components are fixed except for the s th, which undergoes a unit change. This definition implies that a positive step along p increases x_{ν_s} but leaves all the other nonbasics unchanged. The required direction is defined by the equality constrained QP subproblem:

$$\underset{p}{\text{minimize}} \quad \hat{g}(x)^T p + \frac{1}{2} p^T H p \quad \text{subject to} \quad Ap = 0, \quad p_N = e_s, \quad (\text{A.3})$$

and is said to be *nonbinding* with respect to the nonbasic variables. If the multipliers for the constraints $Ap = 0$ are defined in terms of an increment q_π to π , then p_B and q_π satisfy the optimality conditions

$$\left(\begin{array}{cc|c} H_B & -A_B^T & H_D \\ A_B & 0 & A_N \\ \hline 0 & 0 & I_N \end{array} \right) \begin{pmatrix} p_B \\ q_\pi \\ p_N \end{pmatrix} = - \begin{pmatrix} \widehat{g}_B(x) - A_B^T \pi \\ 0 \\ -e_s \end{pmatrix},$$

where, as above, $\widehat{g}_B(x)$ are the basic components of $\widehat{g}(x)$, and H_B and H_D are the basic rows of the basic and nonbasic columns of H . If x is a subspace minimizer, then $\widehat{g}_B(x) - A_B^T \pi = 0$, so that this system simplifies to

$$\left(\begin{array}{cc|c} H_B & -A_B^T & H_D \\ A_B & 0 & A_N \\ \hline 0 & 0 & I_N \end{array} \right) \begin{pmatrix} p_B \\ q_\pi \\ p_N \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ e_s \end{pmatrix}, \quad (\text{A.4})$$

yielding p_B and q_π as the solution of the smaller system

$$\begin{pmatrix} H_B & -A_B^T \\ A_B & 0 \end{pmatrix} \begin{pmatrix} p_B \\ q_\pi \end{pmatrix} = - \begin{pmatrix} (h_{\nu_s})_B \\ a_{\nu_s} \end{pmatrix}. \quad (\text{A.5})$$

The increment q_N for multipliers z_N are computed from p_B , p_N and q_π as $q_N = (Hp - A^T q_\pi)_N$. Once p_B and q_π are known, a nonnegative step α is computed so that $x + \alpha p$ is feasible and $\widehat{f}(x + \alpha p) \leq \widehat{f}(x)$. The step that minimizes \widehat{f} as a function of α is given by

$$\alpha_* = \begin{cases} -\widehat{g}(x)^T p / p^T H p & \text{if } p^T H p > 0, \\ +\infty & \text{otherwise.} \end{cases} \quad (\text{A.6})$$

The best feasible step is then $\alpha = \min\{\alpha_*, \alpha_M\}$, where α_M is the maximum feasible step:

$$\alpha_M = \min_{1 \leq i \leq n_B} \{\gamma_i\}, \quad \text{where } \gamma_i = \begin{cases} \frac{(x_B)_i}{-(p_B)_i} & \text{if } (p_B)_i < 0, \\ +\infty & \text{otherwise.} \end{cases} \quad (\text{A.7})$$

(As $p_N = e_s$ and the problem contains only lower bounds, $x + tp$ remains feasible with respect to the nonbasic variables for all $t \geq 0$.) If $\alpha = +\infty$ then \widehat{f} decreases without limit along p and the problem is unbounded. Otherwise, the new iterate is $(\bar{x}, \bar{\pi}) = (x + \alpha p, \pi + \alpha q_\pi)$.

It is instructive to define the step α_* of (A.6) in terms of the identities

$$\widehat{g}(x)^T p = z_{\nu_s} \quad \text{and} \quad p^T H p = (q_N)_s, \quad (\text{A.8})$$

which follow from the equations (A.4) that define p_B and p_N . Then, if α_* is bounded, we have $\alpha_* = -z_{\nu_s}/(q_N)_s$, or, equivalently,

$$z_{\nu_s} + \alpha_*(q_N)_s = 0.$$

Let $z(t)$ denote the vector of reduced costs at any point on the ray $(x + tp, \pi + tq_\pi)$, i.e., $z(t) = \hat{g}(x+tp) - A^T(\pi + tq_\pi)$. It follows from the definition of p and q_π of (A.4) that $z_B(t) = 0$ for all t , which implies that $x + tp$ is a subspace stationary point *for any step t*. (Moreover, $x + tp$ is a subspace minimizer because the KKT matrix K_B is independent of t .) This property, known as the *parallel subspace property of quadratic programming*, implies that $x + tp$ is the solution of an equality-constraint QP in which the bound on the s th nonbasic is *shifted* to pass through $x + tp$. The component $z_{\nu_s}(t)$ is the reduced cost associated with the shifted version of the bound $x_{\nu_s} \geq 0$. By definition, the s th nonbasic reduced cost is negative at x , i.e., $z_{\nu_s}(0) < 0$. Moreover, a simple calculation shows that $z_{\nu_s}(t)$ is an increasing linear function of t with $z_{\nu_s}(\alpha_*) = 0$ if α_* is bounded. A zero reduced cost at $t = \alpha_*$ means that the shifted bound can be removed from the equality-constraint problem (A.3) (defined at $x = \bar{x}$) without changing its minimizer. Hence, if $\bar{x} = x + \alpha_* p$, the index ν_s is moved to the basic set, which adds column a_{ν_s} to A_B for the next iteration. The shifted variable has been removed from the nonbasic set, which implies that $(\bar{x}, \bar{\pi})$ is a *standard* subspace minimizer.

If we take a shorter step to the boundary of the feasible region, i.e., $\alpha_M < \alpha_*$, then at least one basic variable lies on its bound at $\bar{x} = x + \alpha p$, and one of these, x_{β_r} say, is made nonbasic. If \bar{A}_B denotes the matrix A_B with column r deleted, then \bar{A}_B is not guaranteed to have full row rank (for example, if x is a vertex, A_B is square and \bar{A}_B has more rows than columns). The linear independence of the rows of \bar{A}_B is characterized by the so-called “singularity vector” u_B given by the solution of the equations

$$\begin{pmatrix} H_B & -A_B^T \\ A_B & 0 \end{pmatrix} \begin{pmatrix} u_B \\ v_\pi \end{pmatrix} = \begin{pmatrix} e_r \\ 0 \end{pmatrix}. \quad (\text{A.9})$$

The matrix \bar{A}_B has full rank if and only if $u_B \neq 0$. If \bar{A}_B is rank deficient, \bar{x} is a subspace minimizer with respect to the basis defined by removing x_{ν_s} , i.e., x_{ν_s} is effectively replaced by x_{β_r} in the nonbasic set. In this case, it is necessary to update the dual variables again to reflect the change of basis (see Gill and Wong [96] for more details). The new multipliers are $\bar{\pi} + \sigma v_\pi$, where $\sigma = \hat{g}(\bar{x})^T p / (p_B)_r$.

As defined above, this method requires the solution of two KKT systems at each step (i.e., equations (A.5) and (A.9)). However, if the solution of (A.9) is such that $u_B \neq 0$, then the vectors p_B and q_π needed at \bar{x} can be updated in $O(n)$ operations using the vectors u_B and v_π . Hence, it is unnecessary to solve (A.5) when a basic variable is removed from \mathcal{B} following a restricted step.

Given an initial standard subspace minimizer x_0 and basic set \mathcal{B}_0 , this procedure generates a sequence of primal-dual iterates $\{(x_j, \pi_j)\}$ and an associated sequence of basic sets $\{\mathcal{B}_j\}$. The iterates occur in groups of consecutive iterates that start and end at a standard subspace minimizer.

Each of the intermediate iterates is a nonstandard subspace minimizer at which the same nonbasic variable may not be on its bound. At each intermediate iterate, a variable moves from \mathcal{B} to \mathcal{N} . At the first (standard) subspace minimizer of the group, a nonbasic variable with a negative reduced cost is targeted for inclusion in the basic set. In the subsequent set of iterations, this reduced cost is nondecreasing and the number of basic variables decreases. The group of consecutive iterates ends when the targeted reduced cost reaches zero, at which point the associated variable is made basic.

The method outlined above is based on a method first defined for constraints in all-inequality form by Fletcher [60], and extended to sparse QP by Gould [103]. Recent refinements, including the technique for reducing the number of KKT solves, are given by Gill and Wong [96]. Each of these methods is an example of an *inertia-controlling method*. The idea of an inertia-controlling method is to use the active-set strategy to limit the number of zero and negative eigenvalues in the KKT matrix K_B so that it has inertia $(n_B, m, 0)$ (for a survey, see Gill et al. [92]). At an arbitrary feasible point, a subspace minimizer can be defined by making sufficiently many variables temporarily nonbasic at their current value (see, e.g., Gill, Murray and Saunders [83] for more details).

A.1.2. Binding-direction methods. The next method employs a more conventional active-set strategy in which the nonbasic variables are always active. We start by assuming that the QP is *strictly convex*, i.e., that H is positive definite. Suppose that (x, π) is a feasible primal-dual pair such that $x_i = 0$ for $i \in \mathcal{N}$, where \mathcal{N} is chosen so that A_B has rank m . As in a nonbinding direction method, the primal-dual direction (p, q_π) is computed from an equality constrained QP subproblem. However, in this case the constraints of the subproblem not only force $Ap = 0$ but also require that *every* nonbasic variable remains unchanged for steps of the form $x + \alpha p$. This is done by fixing the nonbasic components of p at zero, giving the equality constraints $Ap = A_B p_B + A_N p_N = 0$ and $p_N = 0$. The resulting subproblem defines a direction that is *binding*, in the sense that it is “bound” or “attached” to the constraints in the nonbasic set. The QP subproblem that gives the best improvement in \hat{f} is then

$$\underset{p}{\text{minimize}} \quad \hat{g}(x)^T p + \frac{1}{2} p^T H p \quad \text{subject to} \quad A_B p_B = 0, \quad p_N = 0. \quad (\text{A.10})$$

The optimality conditions imply that p_B and q_π satisfy the KKT system

$$\begin{pmatrix} H_B & -A_B^T \\ A_B & 0 \end{pmatrix} \begin{pmatrix} p_B \\ q_\pi \end{pmatrix} = - \begin{pmatrix} \hat{g}_B(x) - A_B^T \pi \\ 0 \end{pmatrix}. \quad (\text{A.11})$$

These equations are nonsingular under our assumptions that H is positive definite and A_B has rank m . If (x, π) is a subspace stationary point, then $z_B = \hat{g}_B(x) - A_B^T \pi = 0$ and the solution (p_B, q_π) is zero. In this case, no

improvement can be made in \hat{f} along directions in the null-space of A_B . If the components of $z = \hat{g}(x) - A^T\pi$ are nonnegative then x is optimal for (A.1). Otherwise, a nonbasic variable with a negative reduced cost is selected and moved to the basic set (with no change to x), thereby defining (A.11) with new A_B , H_B and (necessarily nonzero) right-hand side. Given a nonzero solution of (A.11), $x + p$ is either feasible or infeasible with respect to the bounds. If $x + p$ is infeasible, \mathcal{N} cannot be the correct nonbasic set and feasibility is maintained by limiting the step by the maximum feasible step α_M as in (A.7). At the point $\bar{x} = x + \alpha p$, at least one of the basic variables must reach its bound and it is moved to the nonbasic set for the next iteration. Alternatively, if $x + p$ is feasible, $\bar{x} = x + p$ is a subspace minimizer and a nonoptimal nonbasic variable is made basic as above.

The method described above defines groups of consecutive iterates that start with a variable being made basic. No more variables are made basic until either an unconstrained step is taken (i.e., $\alpha = 1$), or a sequence of constrained steps results in the definition of a subspace minimizer (e.g., at a vertex). At each constrained step, the number of basic variables decreases.

As H is positive definite in the strictly convex case, the KKT equations (A.11) remain nonsingular as long as A_B has rank m . One of the most important properties of a binding-direction method is that once an initial nonbasic set is chosen (with the implicit requirement that the associated A_B has rank m), then all subsequent A_B will have rank m (and hence the solution of the KKT system is always well defined). This result is of sufficient importance that we provide a brief proof.

If a variable becomes basic, a column is added to A_B and the rank does not change. It follows that the only possibility for A_B to lose rank is when a basic variable is made nonbasic. Assume that A_B has rank m and that the *first* basic variable is selected to become nonbasic, i.e., $r = 1$. If \bar{A}_B denotes the matrix A_B without its first column, then $A_B = (a_{\beta_r} \quad \bar{A}_B)$. If \bar{A}_B does not have rank m then there must exist a nonzero m -vector \bar{v} such that $\bar{A}_B^T \bar{v} = 0$. If σ denotes the quantity $\sigma = -a_{\beta_r}^T \bar{v}$, then the $(m+1)$ -vector $v = (\bar{v}, \sigma)$ satisfies

$$\begin{pmatrix} a_{\beta_r}^T & 1 \\ \bar{A}_B^T & 0 \end{pmatrix} \begin{pmatrix} \bar{v} \\ \sigma \end{pmatrix} = 0, \text{ or equivalently, } (A_B^T \quad e_r) v = 0.$$

The scalar σ must be nonzero or else $A_B^T \bar{v} = 0$, which would contradict the assumption that A_B has rank m . Then

$$v^T \begin{pmatrix} A_B \\ e_r^T \end{pmatrix} p_B = v^T \begin{pmatrix} 0 \\ (p_B)_r \end{pmatrix} = \sigma(p_B)_r = 0,$$

which implies that $(p_B)_r = 0$. This is a contradiction because the ratio test (A.7) will choose β_r as the outgoing basic variable only if $(p_B)_r < 0$. It follows that $\bar{v} = 0$, and hence \bar{A}_B must have rank m .

If H is not positive definite, the KKT matrix K_B associated with the equations (A.11) may have fewer than n_B positive eigenvalues (cf. (A.2)), i.e., the reduced Hessian $Z_B^T H_B Z_B$ may be singular or indefinite. In this situation, the subproblem (A.10) is unbounded and the equations (A.11) cannot be used directly to define p . In this case we seek a direction p such that $p_N = 0$ and $A_B p_B = 0$, where

$$g_B^T p_B < 0, \quad \text{and} \quad p_B^T H_B p_B \leq 0. \quad (\text{A.12})$$

The QP objective decreases without bound along such a direction, so either the largest feasible step α_M (A.7) is infinite, or a basic variable must become nonbasic at some finite α_M such that $\hat{f}(x + \alpha_M p) \leq \hat{f}(x)$. If $\alpha_M = +\infty$, the QP problem is unbounded and the algorithm is terminated.

A number of methods¹ maintain an unsymmetric block-triangular decomposition of K_B in which the reduced Hessian $Z_B^T H_B Z_B$ is one of the diagonal blocks (the precise form of the decomposition is discussed in Section A.4.1). Given this block-triangular decomposition, the methods of Gill and Murray [82], Gill et al. [87, 92], and Gill, Murray and Saunders [83] factor the reduced Hessian as $L_B D_B L_B^T$, where L_B is unit lower triangular and D_B is diagonal. These methods control the inertia of K_B by starting the iterations at a subspace minimizer. With this restriction, the reduced Hessian has at most one nonpositive eigenvalue, and the direction p_B is unique up to a scalar multiple. This property allows the computation to be arranged so that D_B has at most one nonpositive element, which always occurs in the last position. The vector p_B is then computed from a triangular system involving the rows and columns of L_B associated with the positive-definite principal submatrix of $Z_B^T H_B Z_B$ (see, e.g., Gill et al. [87, 92] for further details).

The method of Bunch and Kaufman [19] allows the reduced Hessian to have any number of nonpositive eigenvalues in the KKT matrix (and therefore need not be started at a subspace minimizer). In this case, a symmetric indefinite factorization of $Z_B^T H_B Z_B$ is maintained, giving a block diagonal factor D_B with 1×1 or 2×2 diagonal blocks. In the strictly convex case, methods may be defined that employ a *symmetric* block decomposition of K_B , see, e.g., Gill et al. [79].

As the reduced Hessian may not be positive definite, methods that maintain a block-triangular decomposition of K_B must use customized methods to factor and modify $Z_B^T H_B Z_B$ as the iterations proceed. This makes it difficult to apply general-purpose solvers that exploit structure in A and H . Methods that factor the KKT system directly are also problematic because K_B can be singular. Fletcher [60] proposed that any potential singularity be handled by embedding K_B in a larger system that is

¹Some were first proposed for the all-inequality constraint case, but they are easily reformulated for constraints in standard form.

known to be nonsingular. This idea was extended to sparse KKT equations by Gould [103]. Fletcher and Gould define an inertia-controlling method based on solving a nonsingular bordered system that includes information associated with the variable x_{β_s} that was most recently made basic. The required binding direction p_B may be found by solving the bordered system

$$\begin{pmatrix} H_B & A_B^T & e_s \\ A_B & & \\ e_s^T & & \end{pmatrix} \begin{pmatrix} p_B \\ -q_\pi \\ -\mu \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

which is nonsingular. A simple calculation shows that p_B , q_π and μ satisfy

$$\begin{pmatrix} \bar{H}_B & -\bar{A}_B^T \\ \bar{A}_B & 0 \end{pmatrix} \begin{pmatrix} p_B \\ q_\pi \end{pmatrix} = - \begin{pmatrix} (h_{\beta_s})_{\bar{\mathcal{B}}} \\ a_{\beta_s} \end{pmatrix} \quad \text{and} \quad \mu = p_B^T H_B p_B, \quad (\text{A.13})$$

where $\bar{\mathcal{B}}$ is the basic set with index β_s omitted. A comparison of (A.13) with (A.5) shows that their respective values of (p_B, q_π) are the same, which implies that Fletcher-Gould binding direction is identical to the nonbinding direction of Section A.1.1. In fact, all binding and nonbinding direction inertia-controlling methods generate the same sequence of iterates when started at the same subspace minimizer. The only difference is in the order in which the computations are performed—binding-direction methods make the targeted nonbasic variable basic at the start of the sequence of consecutive iterates, whereas nonbinding-direction methods make the variable basic at the end of the sequence when the associated shifted bound constraint has a zero multiplier. However, it must be emphasized that not all QP methods are inertia controlling. Some methods allow any number of zero eigenvalues in the KKT matrix—see, for example, the Bunch-Kaufman method mentioned above, and the QP methods in the GALAHAD software package of Gould, Orban, and Toint [109, 110, 104].

A.2. Dual active-set methods. In the convex case (i.e., when H is positive semidefinite) the dual of the QP subproblem (A.1) is

$$\begin{aligned} \underset{w \in \mathbb{R}^n, \pi \in \mathbb{R}^m, z \in \mathbb{R}^n}{\text{minimize}} \quad & \hat{f}_D(w, \pi, z) = b^T \pi + x_I^T z + \frac{1}{2} (w - x_I)^T H (w - x_I) \\ \text{subject to} \quad & H(w - x_I) - A^T \pi - z = -g, \quad z \geq 0. \end{aligned} \quad (\text{A.14})$$

The dual constraints are in standard form, with nonnegativity constraints on z . The optimality conditions for the dual are characterized by the following result.

RESULT A.3 (Optimality conditions for the dual). *The point (w, π, z) is a minimizer of the dual QP (A.14) if and only if*

- (a) (w, π, z) satisfies $H(w - x_I) - A^T \pi - z = -g$, with $z \geq 0$;
- (b) there exists an n -vector y such that
 - (i) $H(w - x_I) = H(y - x_I)$;

- (ii) $Ay = Ax_I - b$, with $y \geq 0$; and
- (iii) $y \cdot z = 0$. □

The components of y are the Lagrange multipliers for the dual bounds $z \geq 0$. Similarly, the components of $y - x_I$ are the “ π -values”, i.e., the multipliers for the equality constraints $H(w - x_I) - A^T\pi - z = -g$. The relationship between the primal and dual solution was first given by Dorn [50]. If the dual has a bounded solution, then part (b) implies that the vector y of multipliers for the dual is a KKT point of the primal, and hence constitutes a primal solution. Moreover, if the dual has a bounded solution and H is positive definite, then $w = y$.

A.2.1. A dual nonbinding direction method. Dual active-set methods can be defined that are based on applying conventional primal active-set methods to the dual problem (A.14). For brevity, we consider the case where H is positive definite; the positive semidefinite case is considered by Gill and Wong [96]. Consider a feasible point (w, π, z) for the dual QP (A.14), i.e., $H(w - x_I) - A^T\pi - z = -g$ and $z \geq 0$. Our intention is to make the notation associated with the dual algorithm consistent with the notation for the primal. To do this, we break with the notation of Section A.1 and use \mathcal{B} to denote the *nonbasic* set and \mathcal{N} to denote the *basic* set for the dual QP. This implies that the dual *nonbasic* variables are $\{z_{\beta_1}, z_{\beta_2}, \dots, z_{\beta_{n_B}}\}$, where $n_B = n - n_N$.

A dual basis contains all the columns of $(H \quad -A^T)$ together with the unit columns corresponding to the dual basic variables, i.e., the columns of I with indices in \mathcal{N} . It follows that the rows and columns of the dual basis may be permuted to give

$$\begin{pmatrix} H_{\mathcal{B}} & H_{\mathcal{D}} & -A_{\mathcal{B}}^T & 0 \\ H_{\mathcal{D}}^T & H_{\mathcal{N}} & -A_{\mathcal{N}}^T & -I_{\mathcal{N}} \end{pmatrix}, \quad (\text{A.15})$$

where $A_{\mathcal{N}}$ and $A_{\mathcal{B}}$ denote the columns of A indexed by \mathcal{N} and \mathcal{B} . The dual nonbasic set $\mathcal{B} = \{\beta_1, \beta_2, \dots, \beta_{n_B}\}$ now provides an estimate of which of the bounds $z \geq 0$ are active at the solution of (A.14). As H is positive definite, the dual basis has full row rank regardless of the rank of the submatrix $-A_{\mathcal{B}}^T$. This implies that if the columns of $A_{\mathcal{B}}$ are to be used to define a basis for a primal solution, it is necessary to impose additional conditions on the dual basis. Here, we assume that the matrix

$$K_{\mathcal{B}} = \begin{pmatrix} H_{\mathcal{B}} & A_{\mathcal{B}}^T \\ A_{\mathcal{B}} & 0 \end{pmatrix} \quad (\text{A.16})$$

is nonsingular. This condition ensures that $A_{\mathcal{B}}$ has rank m . To distinguish $K_{\mathcal{B}}$ from the full KKT matrix for the dual, we refer to $K_{\mathcal{B}}$ as the *reduced KKT matrix*. The next result concerns the properties of a subspace minimizer for the dual QP.

RESULT A.4 (Properties of a subspace minimizer for the dual). *Consider the dual QP (A.14) with H positive definite.*

- (a) If (w, π, z) is a subspace stationary point, then there exists a vector x such that

$$Hw = Hx, \text{ with } A_B x_B + A_N x_N = Ax_I - b, \text{ and } x_N = 0.$$

- (b) A dual subspace stationary point at which the reduced KKT matrix (A.16) is nonsingular is a dual subspace minimizer.
(c) If (w, π, z) is a standard subspace minimizer, then $z_B = 0$ and $z_N \geq 0$.

This result implies that $x = w$ at a dual subspace minimizer for the special case of H positive definite. However, it is helpful to distinguish between w and x to emphasize that x is the vector of dual variables for the dual problem. At a subspace stationary point, x is a basic solution of the primal equality constraints. Moreover, $z = H(w - x_I) - A^T\pi + g = \hat{g}(w) - A^T\pi = \hat{g}(x) - A^T\pi$, which are the primal reduced-costs.

Let (w, π) be a nonoptimal dual subspace minimizer for the dual QP (A.14). (It will be shown below that the vector w need not be computed explicitly.) As (w, π) is not optimal, there is at least one negative component of the dual multiplier vector x_B , say x_{β_r} . If we apply the nonbinding-direction method of Section A.1.1, we define a dual search direction $(\Delta w, q_\pi, \Delta z)$ that is feasible for the dual equality constraints and increases a nonbasic variable with a negative multiplier. As (w, π, z) is assumed to be dual feasible, this gives the constraints for the equality-constraint QP subproblem in the form

$$H\Delta w - A^T q_\pi - \Delta z = 0, \quad \Delta z_B = e_r.$$

The equations analogous to (A.4) for the dual direction $(p, q_\pi, \Delta z)$ are

$$\left(\begin{array}{cccccc|c} H_B & H_D & 0 & 0 & -H_B & -H_D & 0 \\ H_D^T & H_N & 0 & 0 & -H_D^T & -H_N & 0 \\ 0 & 0 & 0 & 0 & A_B & A_N & 0 \\ 0 & 0 & 0 & 0 & 0 & I_N & 0 \\ H_B & H_D & -A_B^T & 0 & 0 & 0 & I_B \\ H_D^T & H_N & -A_N^T & -I_N & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & I_B \end{array} \right) \begin{pmatrix} \Delta w_B \\ \Delta w_N \\ q_\pi \\ \Delta z_N \\ p_B \\ p_N \\ \Delta z_B \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ e_r \end{pmatrix},$$

where p_B and p_N denote the changes in the multipliers x of the dual. Block elimination gives $H\Delta w = Hp$, where p_B , p_N and q_π are determined by the equations

$$p_N = 0, \quad \text{and} \quad \begin{pmatrix} H_B & -A_B^T \\ A_B & 0 \end{pmatrix} \begin{pmatrix} p_B \\ q_\pi \end{pmatrix} = \begin{pmatrix} e_r \\ 0 \end{pmatrix}. \quad (\text{A.17})$$

As $H\Delta w = Hp$, the change in z can be computed as $\Delta z = Hp - A^T q_\pi$. The curvature of the dual objective is given by $\Delta w^T H \Delta w = (p_B)_r$ from (A.8).

If the curvature is nonzero, the step $\alpha_* = -(x_B)_r/(p_B)_r$ minimizes the dual objective $\hat{f}_D(w + \alpha\Delta w, \pi + \alpha q_\pi, z + \alpha\Delta z)$ with respect to α , and the r th element of $x_B + \alpha_* p_B$ is zero. If the x_B are interpreted as estimates of the primal variables, the step from x_B to $x_B + \alpha_* p_B$ increases the negative (and hence infeasible) primal variable $(x_B)_r$ until it reaches its bound of zero. If $\alpha = \alpha_*$ gives a feasible point for dual inequalities, i.e., if $z + \alpha_*\Delta z$ are nonnegative, then the new iterate is $(w + \alpha_*\Delta w, \pi + \alpha_* q_\pi, z + \alpha_*\Delta z)$. In this case, the nonbinding variable is removed from the dual nonbasic set, which means that the index β_r is moved to \mathcal{N} and the associated entries of H and A are removed from H_B and A_B .

If $\alpha = \alpha_*$ is unbounded, or $(w + \alpha_*\Delta w, \pi + \alpha_* q_\pi, z + \alpha_*\Delta z)$ is not feasible, the step is the largest α such that $g(w + \alpha\Delta w) - A^T(\pi + \alpha q_\pi)$ is nonnegative. The required value is

$$\alpha_F = \min_{1 \leq i \leq n_N} \{\gamma_i\}, \quad \text{where } \gamma_i = \begin{cases} \frac{(z_N)_i}{-(\Delta z_N)_i} & \text{if } (\Delta z_N)_i < 0 \\ +\infty & \text{otherwise.} \end{cases} \quad (\text{A.18})$$

If $\alpha_F < \alpha_*$ then at least one component of z_N is zero at $(w + \alpha_F\Delta w, \pi + \alpha_F q_\pi, z + \alpha_F\Delta z)$, and the index of one of these, ν_s say, is moved to \mathcal{B} . The composition of the new dual basis is determined by the singularity vector, adapted to the dual QP from the nonbinding direction method of Section A.1.1. Define the vector u , u_π and v such that $u = v - e_{\nu_s}$, where $v_N = 0$, and u_π and v_B are determined by the equations

$$\begin{pmatrix} H_B & -A_B^T \\ A_B & 0 \end{pmatrix} \begin{pmatrix} v_B \\ u_\pi \end{pmatrix} = \begin{pmatrix} (h_{\nu_s})_B \\ a_{\nu_s} \end{pmatrix}.$$

If u is zero, then $(w + \alpha_F q_\pi, \pi + \alpha_F q_\pi, z + \alpha_F\Delta z)$ is a subspace minimizer with respect to the basis defined with variable β_r replaced by constraint ν_s . Otherwise, ν_s is moved to \mathcal{B} , which has the effect of adding the column a_{ν_s} to A_B , and adding a row and column to H_B . As in the primal nonbinding direction method, the vectors p and q_π may be computed in $O(n)$ operations if no column swap is made.

If (w, π, z) is subspace minimizer at which the reduced KKT matrix (A.16) is nonsingular, then the next iterate is also a subspace minimizer with a nonsingular reduced KKT matrix. (For more details, see Gill and Wong [96].)

The algorithm described above is a special case of the method of Gill and Wong [96], which is defined for the general convex case (i.e., when H can be singular). If $H = 0$ this method is equivalent to the dual simplex method. Bartlett and Biegler [5] propose a method for the strictly convex case that uses the Schur-complement method to handle the changes to the KKT equations when the active set changes (see Section A.4.2).

The dual problem (A.14) has fewer inequality constraints than variables, which implies that if H and A have no common nontrivial null vector,

then the dual constraint gradients, the rows of $(H - A^T)$, are linearly independent, and the dual feasible region has no degenerate points. In this situation, an active-set dual method cannot cycle, and will either terminate with an optimal solution or declare the dual problem to be unbounded. This nondegeneracy property does not hold for a dual linear program, but it does hold for strictly convex problems and any QP with H and A of the form

$$H = \begin{pmatrix} \bar{H} & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad A = (\bar{A} \quad -I_m),$$

where \bar{H} is an $(n-m) \times (n-m)$ positive-definite matrix.

A.2.2. Finding an initial dual-feasible point. An initial dual-feasible point may be defined by applying a conventional phase-one method to the dual constraints, i.e., by minimizing the sum of infeasibilities for the dual constraints $H(x - x_I) - A^T\pi - z = -g$, $z \geq 0$. If H is nonsingular and A has full rank, another option is to define $\mathcal{N} = \emptyset$ and compute (x_0, π_0) from the equations

$$\begin{pmatrix} H & -A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ \pi_0 \end{pmatrix} = - \begin{pmatrix} g - Hx_I \\ b - Ax_I \end{pmatrix}.$$

This choice of basis gives $z_0 = 0$, $\hat{c}(x_0) = 0$, with (x_0, π_0, z_0) a dual subspace minimizer.

A.2.3. The Goldfarb-Idnani method. If H is nonsingular, the vectors

$$y = \begin{pmatrix} \pi \\ z \end{pmatrix}, \quad \bar{b} = \begin{pmatrix} b \\ x_I \end{pmatrix}, \quad \text{and} \quad \bar{A} = \begin{pmatrix} A \\ I \end{pmatrix},$$

may be used to eliminate $w - x_I$ from (A.14) to give the dual problem:

$$\underset{y \in \mathbb{R}^{n+m}, z \in \mathbb{R}^n}{\text{minimize}} \quad y^T(\bar{b} - \bar{A}H^{-1}g) + \frac{1}{2}y^T\bar{A}H^{-1}\bar{A}^Ty, \quad \text{subject to } z \geq 0.$$

Some references include: Goldfarb and Idnani [100], Powell [154]. A variant of the Goldfarb and Idnani method for dense convex QP has been proposed by Boland [10].

A.3. QP regularization. The methods considered above rely on the assumption that each basis matrix A_B has rank m . In an active-set method this condition is guaranteed (at least in exact arithmetic) by the active-set strategy if the *initial* basis has rank m . For methods that solve the KKT system by factoring a subset of m columns of A_B (see Section A.4.1), special techniques can be used to select a linearly independent set of m columns from A . These procedures depend on the method used to factor the basis—for example, the SQP code SNOPT employs a combination of LU factorization and basis repair to determine a full-rank basis. If a factorization

reveals that the square submatrix is rank deficient, suspected dependent columns are discarded and replaced by the columns associated with slack variables. However, for methods that solve the KKT system by direct factorization, such as the Schur complement method of Section A.4.2, basis repair is not an option because the factor routine may be a “black-box” that does not incorporate rank-detection. Unfortunately, over the course of many hundreds of iterations, performed with KKT matrices of varying degrees of conditioning, an SQP method can place even the most robust symmetric indefinite solver under considerable stress. (Even a relatively small collection of difficult problems can test the reliability of a solver. Gould, Scott, and Hu [108] report that none of the 9 symmetric indefinite solvers tested was able to solve all of the 61 systems in their collection.) In this situation it is necessary to use a *regularized* method, i.e., a method based on solving equations that are guaranteed to be solvable without the luxury of basis repair.

To illustrate how a problem may be regularized, we start by considering a QP with *equality constraints*, i.e.,

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad g^T(x - x_I) + \frac{1}{2}(x - x_I)^T H(x - x_I) \\ & \text{subject to} \quad Ax = Ax_I - b. \end{aligned} \tag{A.19}$$

Assume for the moment that this subproblem has a feasible primal-dual solution (x^*, π^*) . Given an estimate π_E of the QP multipliers π^* , a positive μ and arbitrary ν , consider the *generalized augmented Lagrangian*

$$\begin{aligned} \mathcal{M}(x, \pi; \pi_E, \mu, \nu) &= \hat{f}(x) - \hat{c}(x)^T \pi_E + \frac{1}{2\mu} \|\hat{c}(x)\|_2^2 \\ &\quad + \frac{\nu}{2\mu} \|\hat{c}(x) + \mu(\pi - \pi_E)\|_2^2 \end{aligned} \tag{A.20}$$

(see Forsgren and Gill [71], and Gill and Robinson [95]). The function \mathcal{M} involves $n + m$ variables and has gradient vector

$$\nabla \mathcal{M}(x, \pi; \pi_E, \mu, \nu) = \begin{pmatrix} \hat{g}(x) - A^T \pi + (1 + \nu) A^T (\pi - \bar{\pi}(x)) \\ \nu \mu (\pi - \bar{\pi}(x)) \end{pmatrix}, \tag{A.21}$$

where $\bar{\pi}(x) = \pi_E - \hat{c}(x)/\mu$. If we happen to know the value of π^* , and define $\pi_E = \pi^*$, then simple substitution in (A.21) shows that (x^*, π^*) is a stationary point of \mathcal{M} for all ν and all positive μ . The Hessian of \mathcal{M} is given by

$$\nabla^2 \mathcal{M}(x, \pi; \pi_E, \mu, \nu) = \begin{pmatrix} H + \left(\frac{1+\nu}{\mu}\right) A^T A & \nu A^T \\ \nu A & \nu \mu I \end{pmatrix}, \tag{A.22}$$

which is independent of π_E . If we make the additional assumptions that ν is *nonnegative* and the reduced Hessian of the QP subproblem is positive definite, then $\nabla^2 \mathcal{M}$ is positive semidefinite for all μ sufficiently small.

Under these assumptions, if $\pi_E = \pi^*$ it follows that (x^*, π^*) is the unique minimizer of the unconstrained problem

$$\underset{x \in \mathbb{R}^n, \pi \in \mathbb{R}^m}{\text{minimize}} \quad \mathcal{M}(x, \pi; \pi_E, \mu, \nu) \quad (\text{A.23})$$

(see, e.g., Gill and Robinson [95], Gill and Wong [96]). This result implies that if π_E is an approximate multiplier vector (e.g., from the previous QP subproblem in the SQP context), then the minimizer of $\mathcal{M}(x, \pi; \pi_E, \mu, \nu)$ will approximate the minimizer of (A.19). In order to distinguish between a solution of (A.19) and a minimizer of (A.23) for an arbitrary π_E , we use (x_*, π_*) to denote a minimizer of $\mathcal{M}(x, \pi; \pi_E, \mu, \nu)$. Observe that stationarity of $\nabla \mathcal{M}$ at (x_*, π_*) implies that $\pi_* = \bar{\pi}(x_*) = \pi_E - \hat{c}(x_*)/\mu$. The components of $\bar{\pi}(x_*)$ are the so-called *first-order multipliers* associated with a minimizer of (A.23).

Particular values of the parameter ν give some well-known functions (although, as noted above, each function defines a problem with the common solution (x_*, π_*)). If $\nu = 0$, then \mathcal{M} is independent of π , with

$$\mathcal{M}(x; \pi_E, \mu) \equiv \mathcal{M}(x; \pi_E, \mu, 0) = \hat{f}(x) - \hat{c}(x)^T \pi_E + \frac{1}{2\mu} \|\hat{c}(x)\|_2^2. \quad (\text{A.24})$$

This is the conventional Hestenes-Powell augmented Lagrangian (1.11) applied to (A.19). If $\nu = 1$ in (A.21), \mathcal{M} is the primal-dual augmented Lagrangian

$$\hat{f}(x) - \hat{c}(x)^T \pi_E + \frac{1}{2\mu} \|\hat{c}(x)\|_2^2 + \frac{1}{2\mu} \|\hat{c}(x) + \mu(\pi - \pi_E)\|_2^2 \quad (\text{A.25})$$

considered by Robinson [155] and Gill and Robinson [95]. If $\nu = -1$, then \mathcal{M} is the proximal-point Lagrangian

$$\hat{f}(x) - \hat{c}(x)^T \pi - \frac{\mu}{2} \|\pi - \pi_E\|_2^2.$$

As ν is negative in this case, $\nabla^2 \mathcal{M}$ is indefinite and \mathcal{M} has an unbounded minimizer. Nevertheless, a unique minimizer of \mathcal{M} for $\nu > 0$ is a saddle-point for an \mathcal{M} defined with a negative ν . Moreover, for $\nu = -1$, (x^*, π^*) solves the min-max problem

$$\min_x \max_{\pi} \hat{f}(x) - \hat{c}(x)^T \pi - \frac{\mu}{2} \|\pi - \pi_E\|_2^2.$$

In what follows, we use $\mathcal{M}(v)$ to denote \mathcal{M} as a function of the primal-dual variables $v = (x, \pi)$ for given values of π_E , μ and ν . Given the initial point $v_I = (x_I, \pi_I)$, the stationary point of $\mathcal{M}(v)$ is $v_* = v_I + \Delta v$, where $\Delta v = (p, q)$ with $\nabla^2 \mathcal{M}(v_I) \Delta v = -\nabla \mathcal{M}(v_I)$. It can be shown that Δv satisfies the equivalent system

$$\begin{pmatrix} H & -A^T \\ A & \mu I \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = - \begin{pmatrix} \hat{g}(x_I) - A^T \pi_I \\ \hat{c}(x_I) + \mu(\pi_I - \pi_E) \end{pmatrix}, \quad (\text{A.26})$$

which is independent of the value of ν (see Gill and Robinson [95]). If $\nu \neq 0$, the primal-dual direction is unique. If $\nu = 0$ (i.e., \mathcal{M} is the conventional augmented Lagrangian (A.24)), Δv satisfies the equations

$$\begin{pmatrix} H & -A^T \\ A & \mu I \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = - \begin{pmatrix} \widehat{g}(x_I) - A^T \pi \\ \widehat{c}(x_I) + \mu(\pi - \pi_E) \end{pmatrix}, \quad (\text{A.27})$$

for an arbitrary vector π . In this case, p is unique but q depends on the choice of π . In particular, if we define the equations (A.27) with $\pi = \pi_I$, then we obtain directions identical to those of (A.26). Clearly, it must hold that p is independent of the choice of ν in (A.21).

The point $(x_*, \pi_*) = (x_I + p, \pi_I + q)$ is the primal-dual solution of the perturbed QP

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad g^T(x - x_I) + \frac{1}{2}(x - x_I)^T H(x - x_I) \\ & \text{subject to} \quad Ax = Ax_I - b - \mu(\pi_* - \pi_E), \end{aligned} \quad (\text{A.28})$$

where the perturbation shifts each constraint of (A.19) by an amount that depends on the corresponding component of $\pi_* - \pi_E$. Observe that the constraint shift depends on the solution, so it cannot be defined *a priori*. The effect of the shift is to regularize the KKT equations by introducing the nonzero (2, 2) block μI . In the regularized case it is not necessary for A to have full row rank for the KKT equations to be nonsingular. A full-rank assumption is required if the (2, 2) block is zero. In particular, if we choose $\pi_E = \pi_I$, the system (A.26) is:

$$\begin{pmatrix} H & -A^T \\ A & \mu I \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = - \begin{pmatrix} \widehat{g}(x_I) - A^T \pi_I \\ \widehat{c}(x_I) \end{pmatrix}. \quad (\text{A.29})$$

These equations define a regularized version of the Newton equations (2.7). These equations also form the basis for the primal-dual formulations of the quadratic penalty method considered by Gould [102] (for related methods, see Murray [141], Biggs [7] and Tapia [166]).

The price paid for the regularized equations is an approximate solution of the original problem. However, once (x_*, π_*) has been found, π_E can be redefined as π_* and the process repeated—with a smaller value of μ if necessary. There is more discussion of the choice of π_E below. However, before turning to the inequality constraint case, we summarize the regularization for equality constraints.

- The primal-dual solution (x^*, π^*) of the equality constraint problem (A.19) is approximated by the solution of the perturbed KKT system (A.26).
- The resulting approximation $(x_*, \pi_*) = (x_I + p, \pi_I + q)$ is a stationary point of the function \mathcal{M} (A.21) for all values of the parameter ν . If $\mu > 0$ and $\nu \geq 0$ then (x_*, π_*) is a minimizer of \mathcal{M} for all μ sufficiently small.

As the solution of the regularized problem is independent of ν , there is little reason to use nonzero values of ν in the equality-constraint case. However, the picture changes when there are *inequality constraints* and an *approximate* solution of the QP problem is required, as is often the case in the SQP context.

The method defined above can be extended to the inequality constraint problem (A.1) by solving, the bound-constrained subproblem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \mathcal{M}(x; \pi_E, \mu) \quad \text{subject to} \quad x \geq 0. \quad (\text{A.30})$$

This technique has been proposed for general nonlinear programming (see, e.g., Conn, Gould and Toint [39, 40, 41], Friedlander [73], and Friedlander and Saunders [75]), and to quadratic programming (see, e.g., Dostál, Friedlander and Santos [51, 52, 53], Delbos and Gilbert [46], Friedlander and Leyffer [74]), and Maes [131]). Subproblem (A.30) may be solved using one of the active-set methods of Sections A.1.2 and A.1.1, although no explicit phase-one procedure is needed because there are no general constraints. In the special case of problem (A.30) a primal active-set method defines a sequence of nonnegative iterates $\{x_j\}$ such that $x_{j+1} = x_j + \alpha_j p_j \geq 0$. At the j th iteration of the binding-direction method of Section A.1.2, variables in the nonbasic set \mathcal{N} remain unchanged for any value of the step length, i.e., $p_N = (p_j)_N = 0$. This implies that the elements p_B of the direction p_j must solve the unconstrained QP subproblem:

$$\underset{p_B}{\text{minimize}} \quad p_B^T (\nabla \mathcal{M}_j)_B + \frac{1}{2} p_B^T (\nabla^2 \mathcal{M}_j)_B p_B.$$

As in (A.26), the optimality conditions imply that p_B satisfies

$$\begin{pmatrix} H_B & -A_B^T \\ A_B & \mu I \end{pmatrix} \begin{pmatrix} p_B \\ q_j \end{pmatrix} = - \begin{pmatrix} (\hat{g}(x_j) - A^T \pi_j)_B \\ \hat{c}(x_j) + \mu(\pi_j - \pi_E) \end{pmatrix}, \quad (\text{A.31})$$

where π_j is an estimate of the optimal multipliers π_* of (A.30). The next iterate is defined as $x_{j+1} = x_j + \alpha_j p_j$, where the steplength α_j is chosen such that $x_j + \alpha_j p_j \geq 0$. As in the equality-constraint case, the dual variables may be updated as $\pi_{j+1} = \pi_j + \alpha_j q_j$. The dual iterates π_j will converge to the multipliers π_* of the perturbed QP:

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad g^T(x - x_I) + \frac{1}{2}(x - x_I)^T H(x - x_I) \\ & \text{subject to} \quad Ax = Ax_I - b - \mu(\pi_* - \pi_E), \quad x \geq 0. \end{aligned} \quad (\text{A.32})$$

At an optimal solution (x_*, π_*) of (A.30) the vector $z_* = \hat{g}(x_*) - A^T \pi_*$ provides an estimate of the optimal reduced costs z^* . As in the equality-constraint case, the vector of first-order multipliers $\bar{\pi}(x_*) = \pi_E - \hat{c}(x_*)/\mu$ is identical to π_* . Problem (A.30) may be solved using a bound-constraint variant of the nonbinding-direction method of Section A.1.1. This method

has some advantages when solving convex and general QP problems. For more details, see Gill and Wong [96].

If the QP is a “one-off” problem, then established techniques associated with the bound-constrained augmented Lagrangian method can be used to update π_E and μ (see, e.g., Conn, Gould and Toint [40], Dostál, Friedlander and Santos [51, 52, 53], Delbos and Gilbert [46], and Friedlander and Leyffer [74]). These rules are designed to update π_E and μ without the need to find the exact solution of (A.30). In the SQP context, it may be more appropriate to find an approximate solution of (A.30) for a *fixed* value of π_E , which is then updated in the outer iteration. Moreover, as μ is being used principally for regularization, it is given a smaller value than is typical in a conventional augmented Lagrangian method.

A.4. Solving the KKT system. The principal work associated with a QP iteration is the cost of solving one or two saddle-point systems of the form

$$\begin{pmatrix} H_B & -A_B^T & H_D \\ A_B & \mu I & A_N \\ & & I_N \end{pmatrix} \begin{pmatrix} y_B \\ w \\ y_N \end{pmatrix} = \begin{pmatrix} g_B \\ f_1 \\ f_2 \end{pmatrix}, \quad (\text{A.33})$$

where μ is a nonnegative scalar. We focus on two approaches appropriate for large-scale quadratic programming.

A.4.1. Variable-reduction methods. These methods are appropriate for the case $\mu = 0$. As A_B has rank m , there exists a nonsingular Q_B such that

$$A_B Q_B = (0 \quad B), \quad (\text{A.34})$$

with B an $m \times m$ nonsingular matrix. If $\mu = 0$ the matrix Q_B is used to transform the generic system (A.33) to block-triangular form. The columns of Q_B are partitioned so that $Q_B = (Z_B \quad Y_B)$ with Z_B an $n_B - m$ by n_B matrix, then $A_B Z_B = 0$ and the columns of Z_B span the null-space of A_B . Analogous to (2.11) we obtain the permuted block-triangular system:

$$\begin{pmatrix} Z_B^T H_B Z_B & Z_B^T H_B Y_B & Z_B^T H_D \\ Y_B^T H_B Z_B & Y_B^T H_B Y_B & -B^T \\ & B & 0 \end{pmatrix} \begin{pmatrix} y_Z \\ y_Y \\ w \end{pmatrix} = \begin{pmatrix} g_Z \\ g_Y \\ f_1 \end{pmatrix}, \quad (\text{A.35})$$

with $g_Z = Z_B^T g_B$ and $g_Y = Y_B^T g_B$. We formulate the result of the block substitution in a form that uses matrix-vector products involving the full matrix H rather than the submatrix H_B . This is done to emphasize the practical utility of accessing the QP Hessian as an *operator* that defines the product Hx for a given x . This reformulation requires the definition of the explicit column permutation P that identifies the basic and nonbasic columns A_B and A_N of A , i.e.,

$$AP = (A_B \quad A_N). \quad (\text{A.36})$$

Given the permutation P , we define matrices Q , W^T , Y^T and Z^T that act on vectors of length n , i.e., $Q = (Z \ Y \ W)$, where

$$Z = P \begin{pmatrix} Z_B \\ 0 \end{pmatrix}, \quad Y = P \begin{pmatrix} Y_B \\ 0 \end{pmatrix}, \quad \text{and} \quad W = P \begin{pmatrix} 0 \\ I_N \end{pmatrix}.$$

In terms of these matrices, the block substitution yields

$$\begin{aligned} y_W &= f_2, & y_0 &= Wy_W, \\ By_Y &= f_1 - A_N f_2, & y_1 &= Yy_Y + y_0, \\ Z^T H Z y_Z &= Z^T(g - Hy_1), & y &= Zy_Z + y_1, \\ B^T w &= -Y^T(g - Hy). \end{aligned} \quad (\text{A.37})$$

There are many practical choices for the matrix Q_B . For small-to-medium scale problems with dense A and H , the matrix Q_B can be calculated as the orthogonal factor associated with the QR factorization of a row and column-permuted A_B^T (see, e.g., Gill et al. [87]). The method of *variable reduction* is appropriate when A is sparse. In this case the permutation P of (A.36) is specialized further to give

$$AP = (A_B \ A_N), \quad \text{with} \quad A_B = (B \ S),$$

where B an $m \times m$ nonsingular subset of the columns of A and S an $m \times n_S$ matrix with $n_S = n_B - m$. The matrix $Q = (Z \ Y \ W)$ is constructed so that

$$Z = P \begin{pmatrix} -B^{-1}S \\ I_{n_S} \\ 0 \end{pmatrix}, \quad Y = P \begin{pmatrix} I_m \\ 0 \\ 0 \end{pmatrix}, \quad \text{and} \quad W = P \begin{pmatrix} 0 \\ 0 \\ I_N \end{pmatrix}.$$

This form means that matrix-vector products $Z^T v$ or Zv can be computed using a factorization of B (typically, a sparse LU factorization; see Gill et al. [89]), and Z need not be stored explicitly.

A.4.2. The Schur complement method. Solving a “one-off” KKT system can be done very effectively using sparse matrix factorization techniques. However, within a QP algorithm, many systems must be solved in which the matrix changes by a single row and column. Instead of redefining and re-solving the KKT equations at each iteration, the solution may be found by solving a bordered system of the form

$$\begin{pmatrix} K_0 & V \\ V^T & D \end{pmatrix} \begin{pmatrix} z \\ w \end{pmatrix} = \begin{pmatrix} b \\ f \end{pmatrix}, \quad \text{with} \quad K_0 = \begin{pmatrix} H_B & A_B^T \\ A_B & \mu I \end{pmatrix}, \quad (\text{A.38})$$

where K_0 is the KKT matrix at the initial point. For simplicity, we assume that the second block of the variables is scaled by -1 so that the $(1, 2)$ block of K_0 is A_B^T , not $-A_B^T$. The *Schur complement method* is based on the assumption that factorizations for K_0 and the *Schur complement*

$C = D - V^T K_0^{-1} V$ exist. Then the solution of (A.38) can be determined by solving the equations

$$K_0 t = b, \quad Cw = f - V^T t, \quad K_0 z = b - Vw.$$

The work required is dominated by two solves with the fixed matrix K_0 and one solve with the Schur complement C . If the number of changes to the basic set is small enough, dense factors of C may be maintained.

We illustrate the definition of (A.38) immediately after the matrix K_0 is factorized. (For more details, see, e.g., Bisschop and Meerbeau [8], Gill et al. [91].) Suppose that variable s enters the basic set. The next KKT matrix can be written as

$$\left(\begin{array}{cc|c} H_B & A_B^T & (h_s)_B \\ A_B & \mu I & a_s \\ \hline (h_s)_B^T & a_s^T & h_{ss} \end{array} \right),$$

where a_s and h_s are the s th columns of A and H . This is a matrix of the form (A.38) with $D = (h_{ss})$ and $V^T = ((h_s)_B^T \ a_s^T)$.

Now consider the case where the r th basic variable is deleted from the basic set, so that the r th column is removed from A_B . The corresponding changes can be enforced in the solution of the KKT system using the bordered matrix:

$$\left(\begin{array}{ccc|c} H_B & A_B^T & (h_s)_B & e_r \\ A_B & \mu I & a_s & 0 \\ (h_s)_B^T & a_s^T & h_{ss} & 0 \\ \hline e_r^T & 0 & 0 & 0 \end{array} \right).$$

Bordering with the unit row and column has the effect of zeroing out the components of the solution corresponding to the deleted basic variable.

The Schur complement method can be extended to a *block LU method* by storing the bordered matrix in block-factored form

$$\begin{pmatrix} K_0 & V \\ V^T & D \end{pmatrix} = \begin{pmatrix} L & \\ Z^T & I \end{pmatrix} \begin{pmatrix} U & Y \\ & C \end{pmatrix}, \quad (\text{A.39})$$

where $K_0 = LU$, $LY = V$, $U^T Z = V$, and $C = D - Z^T Y$, which is the Schur complement matrix (see Eldersveld and Saunders [56], Huynh [123]).

Using the block factors, the solution of (A.38) can be computed from the equations

$$Lt = b, \quad Cw = f - Z^T t, \quad Uz = t - Yw.$$

This method requires a solve with L and U each, one multiply with Y and Z^T , and one solve with the Schur complement C .

Although the augmented system (in general) increases in dimension by one at every iteration, the K_0 block is fixed and defined by the initial basic set. As the inner iterations proceed, the size of C increases and the work required to perform a solve or update increases. It may be necessary to restart the process by discarding the existing factors and re-forming K_0 based on the current set of basic variables.

A.5. Finding an initial feasible point. There are two approaches to finding a feasible point for the QP constraints. The first, common in linear programming, is to find a point that satisfies the equality constraints and then iterate (if necessary) to satisfy the bounds. The second method finds a point that satisfies the bound constraints and then iterates to satisfy the equality constraints. In each case we assume that an initial nonbasic set is known (in the SQP context, this is often the final nonbasic set from the previous QP subproblem).

The first approach is suitable if the variable reduction method is used in the optimality phase. In this case, a factorization is available of the matrix B such that $A_B Q_B = (0 \quad B)$. Given x_I , a point x_0 is computed that satisfies the constraints $Ax = Ax_I - b$, i.e., we define

$$Bp_Y = -\hat{c}(x_I), \quad p_F = Yp_Y, \quad x_0 = x_I + p_F.$$

If $x_0 \not\geq 0$, then x_0 is the initial iterate for a phase-one algorithm that minimizes the linear function $-\sum_{i \in \mathcal{V}(x)} x_i$, where $\mathcal{V}(x)$ is the index set of violated bounds at x . The two-phase nature of the algorithm is reflected by changing the function being minimized from the sum of infeasibilities to the quadratic objective function. The function $-\sum_{i \in \mathcal{V}(x)} x_i$ is a piece-wise linear function that gives the one-norm of the constraint infeasibilities at x . A feature of this approach is that many violated constraints can become feasible at any given iteration.

Minimizing the explicit sum of infeasibilities directly is less straightforward for the Schur complement method because the objective function changes from one iteration to the next. In this case, a single phase with a composite objective may be used, i.e.,

$$\begin{aligned} & \underset{x, v, w}{\text{minimize}} \quad g^T(x - x_I) + \frac{1}{2}(x - x_I)^T H(x - x_I) + e^T u + e^T v \\ & \text{subject to} \quad Ax - u + v = Ax_I - b, \quad x \geq 0, \quad u \geq 0, \quad v \geq 0, \end{aligned} \tag{A.40}$$

where e is the vector of ones. This approach has been used by Gould [103], and Huynh [123]. An alternative is to define a phase-one subproblem that minimizes the two-norm of the constraint violations, i.e.,

$$\underset{x, v}{\text{minimize}} \quad \frac{1}{2}\|v\|_2^2 \quad \text{subject to} \quad Ax + v = Ax_I - b, \quad x \geq 0. \tag{A.41}$$

This problem is a convex QP. Given an initial point x_0 and nonbasic set \mathcal{N}_0 for the phase-two problem, the basic variables for phase one consist of

the x_0 variables in \mathcal{B}_0 and the m variables v_0 such that $Ax_0 + v_0 = Ax_I - b$. The variables v are always basic.

Another phase-one subproblem appropriate for the Schur complement method minimizes a strictly convex objective involving the two-norm of the constraint violations and a primal regularization term:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \|Ax - b\|_2^2 + \frac{1}{2}\sigma \|x - x_I\|_2^2 \quad \text{subject to } x \geq 0. \quad (\text{A.42})$$

If a feasible point exists, then the problem is feasible with the objective bounded below by zero. The only constraints of this problem are bounds on x . Applying the nonbinding direction method of Section A.1.1 gives

$$\begin{pmatrix} \sigma I_B & A_B^T \\ A_B & -I \end{pmatrix} \begin{pmatrix} p_B \\ -q_\pi \end{pmatrix} = \begin{pmatrix} 0 \\ -a_{\nu_s} \end{pmatrix},$$

with $p_N = e_s$ and $q_N = \sigma e_s - A_N^T q_\pi$. Solving this phase-one problem is equivalent to applying the regularized QP algorithm in [96] with $\mu = 1$ and $\pi_E = 0$, to the problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{\sigma}{2} \|x - x_0\|_2^2 \quad \text{subject to } Ax = b, \quad x \geq 0.$$

REFERENCES

- [1] P.R. AMESTOY, I.S. DUFF, J.-Y. L'EXCELLENT, AND J. KOSTER, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., **23** (2001), pp. 15–41 (electronic).
- [2] M. ANITESCU, *On the rate of convergence of sequential quadratic programming with nondifferentiable exact penalty function in the presence of constraint degeneracy*, Math. Program., **92** (2002), pp. 359–386.
- [3] ———, *A superlinearly convergent sequential quadratically constrained quadratic programming algorithm for degenerate nonlinear programming*, SIAM J. Optim., **12** (2002), pp. 949–978.
- [4] C. ASHCRAFT AND R. GRIMES, *SPOOLES: an object-oriented sparse matrix library*, in Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing 1999 (San Antonio, TX), Philadelphia, PA, 1999, SIAM, p. 10.
- [5] R.A. BARTLETT AND L.T. BIEGLER, *QPSchur: a dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming*, Optim. Eng., **7** (2006), pp. 5–32.
- [6] D.P. BERTSEKAS, *Constrained optimization and Lagrange multiplier methods*, Athena Scientific, Belmont, Massachusetts, 1996.
- [7] M.C. BIGGS, *Constrained minimization using recursive equality quadratic programming*, in Numerical Methods for Nonlinear Optimization, F.A. Lootsma, ed., Academic Press, London and New York, 1972, pp. 411–428.
- [8] J. BISSCHOP AND A. MEERAUS, *Matrix augmentation and partitioning in the updating of the basis inverse*, Math. Program., **13** (1977), pp. 241–254.
- [9] P.T. BOGGS AND J.W. TOLLE, *Sequential quadratic programming*, in Acta Numerica, 1995, Vol. **4** of Acta Numer., Cambridge Univ. Press, Cambridge, 1995, pp. 1–51.

- [10] N.L. BOLAND, *A dual-active-set algorithm for positive semi-definite quadratic programming*, Math. Programming, **78** (1997), pp. 1–27.
- [11] J.M. BORWEIN, *Necessary and sufficient conditions for quadratic minimality*, Numer. Funct. Anal. and Optimiz., **5** (1982), pp. 127–140.
- [12] A.M. BRADLEY, *Algorithms for the Equilibration of Matrices and Their Application to Limited-Memory Quasi-Newton Methods*, PhD thesis, Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA, May 2010.
- [13] K.W. BRODLIE, A.R. GOURLAY, AND J. GREENSTADT, *Rank-one and rank-two corrections to positive definite matrices expressed in product form*, J. Inst. Math. Appl., **11** (1973), pp. 73–82.
- [14] C.G. BROYDEN, *The convergence of a class of double rank minimization algorithms, I & II*, J. Inst. Maths. Applns., **6** (1970), pp. 76–90 and 222–231.
- [15] A. BUCKLEY AND A. LENIR, *QN-like variable storage conjugate gradients*, Math. Program., **27** (1983), pp. 155–175.
- [16] ———, *BBVSCG—a variable storage algorithm for function minimization*, ACM Trans. Math. Software, **11** (1985), pp. 103–119.
- [17] J.R. BUNCH, *Partial pivoting strategies for symmetric matrices*, SIAM J. Numer. Anal., **11** (1974), pp. 521–528.
- [18] J.R. BUNCH AND L. KAUFMAN, *Some stable methods for calculating inertia and solving symmetric linear systems*, Math. Comput., **31** (1977), pp. 163–179.
- [19] ———, *A computational method for the indefinite quadratic programming problem*, Linear Algebra Appl., **34** (1980), pp. 341–370.
- [20] J.R. BUNCH AND B.N. PARLETT, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., **8** (1971), pp. 639–655.
- [21] J.V. BURKE, *A sequential quadratic programming method for potentially infeasible mathematical programs*, J. Math. Anal. Appl., **139** (1989), pp. 319–351.
- [22] ———, *A robust trust region method for constrained nonlinear programming problems*, SIAM J. Optim., **2** (1992), pp. 324–347.
- [23] J.V. BURKE AND S.-P. HAN, *A robust sequential quadratic programming method*, Math. Programming, **43** (1989), pp. 277–303.
- [24] R.H. BYRD, F.E. CURTIS, AND J. NOCEDAL, *Infeasibility detection and SQP methods for nonlinear optimization*, SIAM Journal on Optimization, **20** (2010), pp. 2281–2299.
- [25] R.H. BYRD, N.I.M. GOULD, J. NOCEDAL, AND R.A. WALTZ, *An algorithm for nonlinear optimization using linear programming and equality constrained subproblems*, Math. Program., **100** (2004), pp. 27–48.
- [26] ———, *On the convergence of successive linear-quadratic programming algorithms*, SIAM J. Optim., **16** (2005), pp. 471–489.
- [27] R.H. BYRD, J. NOCEDAL, AND R.B. SCHNABEL, *Representations of quasi-Newton matrices and their use in limited-memory methods*, Math. Program., **63** (1994), pp. 129–156.
- [28] R.H. BYRD, J. NOCEDAL, AND R.A. WALTZ, *Steering exact penalty methods for nonlinear programming*, Optim. Methods Softw., **23** (2008), pp. 197–213.
- [29] R.H. BYRD, R.A. TAPIA, AND Y. ZHANG, *An SQP augmented Lagrangian BFGS algorithm for constrained optimization*, SIAM J. Optim., **20** (1992), pp. 210–241.
- [30] R.M. CHAMBERLAIN, M.J.D. POWELL, C. LEMARECHAL, AND H.C. PEDERSEN, *The watchdog technique for forcing convergence in algorithms for constrained optimization*, Math. Programming Stud. (1982), pp. 1–17. Algorithms for constrained minimization of smooth nonlinear functions.
- [31] C.M. CHIN, *A New Trust Region based SLP Filter Algorithm which uses EQP Active Set Strategy*, PhD thesis, Department of Mathematics, University of Dundee, Scotland, 2001.
- [32] C.M. CHIN AND R. FLETCHER, *On the global convergence of an SLP-filter algorithm that takes EQP steps*, Math. Program., **96** (2003), pp. 161–177.

- [33] C.M. CHIN, A.H.A. RASHID, AND K.M. NOR, *A combined filter line search and trust region method for nonlinear programming*, WSEAS Trans. Math., **5** (2006), pp. 656–662.
- [34] J.W. CHINNECK, *Analyzing infeasible nonlinear programs*, Comput. Optim. Appl., **4** (1995), pp. 167–179.
- [35] ———, *Feasibility and infeasibility in optimization: algorithms and computational methods*, International Series in Operations Research & Management Science, **118**, Springer, New York, 2008.
- [36] T.F. COLEMAN AND A.R. CONN, *On the local convergence of a quasi-Newton method for the nonlinear programming problem*, SIAM J. Numer. Anal., **21** (1984), pp. 775–769.
- [37] T.F. COLEMAN AND A. POTHEN, *The null space problem I. Complexity*, SIAM J. on Algebraic and Discrete Methods, **7** (1986), pp. 527–537.
- [38] T.F. COLEMAN AND D.C. SORENSEN, *A note on the computation of an orthogonal basis for the null space of a matrix*, Math. Program., **29** (1984), pp. 234–242.
- [39] A.R. CONN, N.I.M. GOULD, AND PH. L. TOINT, *Global convergence of a class of trust region algorithms for optimization with simple bounds*, SIAM J. Numer. Anal., **25** (1988), pp. 433–460.
- [40] ———, *A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds*, SIAM J. Numer. Anal., **28** (1991), pp. 545–572.
- [41] ———, *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*, Lecture Notes in Computation Mathematics 17, Springer Verlag, Berlin, Heidelberg, New York, London, Paris and Tokyo, 1992.
- [42] ———, *Trust-Region Methods*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.
- [43] L.B. CONTESSE, *Une caractérisation complète des minima locaux en programmation quadratique*, Numer. Math., **34** (1980), pp. 315–332.
- [44] R.W. COTTLE, G.J. HABETLER, AND C.E. LEMKE, *On classes of copositive matrices*, Linear Algebra Appl., **3** (1970), pp. 295–310.
- [45] Y.-H. DAI AND K. SCHITTOKWSKI, *A sequential quadratic programming algorithm with non-monotone line search*, Pac. J. Optim., **4** (2008), pp. 335–351.
- [46] F. DELBOS AND J.C. GILBERT, *Global linear convergence of an augmented Lagrangian algorithm to solve convex quadratic optimization problems*, J. Convex Anal., **12** (2005), pp. 45–69.
- [47] R.S. DEMBO AND U. TULowitzki, *Sequential truncated quadratic programming methods*, in Numerical optimization, 1984 (Boulder, Colo., 1984), SIAM, Philadelphia, PA, 1985, pp. 83–101.
- [48] J.E. DENNIS, JR. AND R.B. SCHNABEL, *A new derivation of symmetric positive definite secant updates*, in Nonlinear Programming, **4** (Proc. Sympos., Special Interest Group on Math. Programming, Univ. Wisconsin, Madison, Wis., 1980), Academic Press, New York, 1981, pp. 167–199.
- [49] G. DiPILLO AND L. GRIPPO, *A new class of augmented Lagrangians in nonlinear programming*, SIAM J. Control Optim., **17** (1979), pp. 618–628.
- [50] W.S. DORN, *Duality in quadratic programming*, Quart. Appl. Math., **18** (1960/1961), pp. 155–162.
- [51] Z. DOSTÁL, A. FRIEDLANDER, AND S. A. SANTOS, *Adaptive precision control in quadratic programming with simple bounds and/or equalities*, in High performance algorithms and software in nonlinear optimization (Ischia, 1997), Vol. **24** of Appl. Optim., Kluwer Acad. Publ., Dordrecht, 1998, pp. 161–173.
- [52] ———, *Augmented Lagrangians with adaptive precision control for quadratic programming with equality constraints*, Comput. Optim. Appl., **14** (1999), pp. 37–53.
- [53] ———, *Augmented Lagrangians with adaptive precision control for quadratic programming with simple bounds and equality constraints*, SIAM J. Optim., **13** (2003), pp. 1120–1140 (electronic).

- [54] I.S. DUFF, *MA57—a code for the solution of sparse symmetric definite and indefinite systems*, ACM Trans. Math. Software, **30** (2004), pp. 118–144.
- [55] I.S. DUFF AND J.K. REID, *MA27: a set of Fortran subroutines for solving sparse symmetric sets of linear equations*, Tech. Rep. R-10533, Computer Science and Systems Division, AERE Harwell, Oxford, England, 1982.
- [56] S.K. ELDERSVELD AND M.A. SAUNDERS, *A block-LU update for large-scale linear programming*, SIAM J. Matrix Anal. Appl., **13** (1992), pp. 191–201.
- [57] O. EXLER AND K. SCHITTkowski, *A trust region SQP algorithm for mixed-integer nonlinear programming*, Optim. Lett., **1** (2007), pp. 269–280.
- [58] A. FISCHER, *Modified Wilson’s method for nonlinear programs with nonunique multipliers*, Math. Oper. Res., **24** (1999), pp. 699–727.
- [59] R. FLETCHER, *A new approach to variable metric algorithms*, Computer Journal, **13** (1970), pp. 317–322.
- [60] ———, *A general quadratic programming algorithm*, J. Inst. Math. Applies., **7** (1971), pp. 76–91.
- [61] ———, *A model algorithm for composite nondifferentiable optimization problems*, Math. Programming Stud. (1982), pp. 67–76. Nondifferential and variational techniques in optimization (Lexington, Ky., 1980).
- [62] ———, *Second order corrections for nondifferentiable optimization*, in Numerical analysis (Dundee, 1981), Vol. **912** of Lecture Notes in Math., Springer, Berlin, 1982, pp. 85–114.
- [63] ———, *An ℓ_1 penalty method for nonlinear constraints*, in Numerical Optimization 1984, P.T. Boggs, R.H. Byrd, and R.B. Schnabel, eds., Philadelphia, 1985, pp. 26–40.
- [64] ———, *Practical methods of optimization*, Wiley-Interscience [John Wiley & Sons], New York, 2001.
- [65] R. FLETCHER, N.I.M. GOULD, S. LEYFFER, PH. L. TOINT, AND A. WÄCHTER, *Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming*, SIAM J. Optim., **13** (2002), pp. 635–659 (electronic) (2003).
- [66] R. FLETCHER AND S. LEYFFER, *User manual for filterSQP*, Tech. Rep. NA/181, Dept. of Mathematics, University of Dundee, Scotland, 1998.
- [67] ———, *Nonlinear programming without a penalty function*, Math. Program., **91** (2002), pp. 239–269.
- [68] R. FLETCHER, S. LEYFFER, AND PH. L. TOINT, *On the global convergence of a filter-SQP algorithm*, SIAM J. Optim., **13** (2002), pp. 44–59 (electronic).
- [69] R. FLETCHER AND E. SAINZ DE LA MAZA, *Nonlinear programming and nonsmooth optimization by successive linear programming*, Math. Program., **43** (1989), pp. 235–256.
- [70] A. FORSGREN, *Inertia-controlling factorizations for optimization algorithms*, Appl. Num. Math., **43** (2002), pp. 91–107.
- [71] A. FORSGREN AND P.E. GILL, *Primal-dual interior methods for nonconvex nonlinear programming*, SIAM J. Optim., **8** (1998), pp. 1132–1152.
- [72] A. FORSGREN, P.E. GILL, AND W. MURRAY, *On the identification of local minimizers in inertia-controlling methods for quadratic programming*, SIAM J. Matrix Anal. Appl., **12** (1991), pp. 730–746.
- [73] M.P. FRIEDLANDER, *A Globally Convergent Linearly Constrained Lagrangian Method for Nonlinear Optimization*, PhD thesis, Department of Operations Research, Stanford University, Stanford, CA, 2002.
- [74] M.P. FRIEDLANDER AND S. LEYFFER, *Global and finite termination of a two-phase augmented Lagrangian filter method for general quadratic programs*, SIAM J. Sci. Comput., **30** (2008), pp. 1706–1729.
- [75] M.P. FRIEDLANDER AND M.A. SAUNDERS, *A globally convergent linearly constrained Lagrangian method for nonlinear optimization*, SIAM J. Optim., **15** (2005), pp. 863–897.

- [76] M.P. FRIEDLANDER AND P. TSENG, *Exact regularization of convex programs*, SIAM J. Optim., **18** (2007), pp. 1326–1350.
- [77] J.C. GILBERT AND C. LEMARÉCHAL, *Some numerical experiments with variable-storage quasi-Newton algorithms*, Math. Program. (1989), pp. 407–435.
- [78] J.R. GILBERT AND M.T. HEATH, *Computing a sparse basis for the null space*, Report TR86-730, Department of Computer Science, Cornell University, 1986.
- [79] P.E. GILL, N.I.M. GOULD, W. MURRAY, M.A. SAUNDERS, AND M.H. WRIGHT, *A weighted gram-schmidt method for convex quadratic programming*, Math. Programming., **30** (1984), pp. 176–195.
- [80] P.E. GILL AND M.W. LEONARD, *Limited-memory reduced-Hessian methods for large-scale unconstrained optimization*, SIAM J. Optim., **14** (2003), pp. 380–401.
- [81] P.E. GILL AND W. MURRAY, *Newton-type methods for unconstrained and linearly constrained optimization*, Math. Program., **7** (1974), pp. 311–350.
- [82] ———, *Numerically stable methods for quadratic programming*, Math. Program., **14** (1978), pp. 349–372.
- [83] P.E. GILL, W. MURRAY, AND M.A. SAUNDERS, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Rev., **47** (2005), pp. 99–131.
- [84] ———, *User’s guide for SNOPT Version 7: Software for large-scale nonlinear programming*, Numerical Analysis Report 06-2, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2006.
- [85] P.E. GILL, W. MURRAY, M.A. SAUNDERS, G.W. STEWART, AND M.H. WRIGHT, *Properties of a representation of a basis for the null space*, Math. Programming, **33** (1985), pp. 172–186.
- [86] P.E. GILL, W. MURRAY, M.A. SAUNDERS, AND M.H. WRIGHT, *A note on a sufficient-decrease criterion for a nonderivative step-length procedure*, Math. Programming, **23** (1982), pp. 349–352.
- [87] ———, *Procedures for optimization problems with a mixture of bounds and general linear constraints*, ACM Trans. Math. Software, **10** (1984), pp. 282–298.
- [88] ———, *Sparse matrix methods in optimization*, SIAM J. Sci. Statist. Comput., **5** (1984), pp. 562–589.
- [89] ———, *Maintaining LU factors of a general sparse matrix*, Linear Algebra Appl., **88/89** (1987), pp. 239–270.
- [90] ———, *A Schur-complement method for sparse quadratic programming*, Report SOL 87-12, Department of Operations Research, Stanford University, Stanford, CA, 1987.
- [91] ———, *A Schur-complement method for sparse quadratic programming*, in Reliable Numerical Computation, M.G. Cox and S.J. Hammarling, eds., Oxford University Press, 1990, pp. 113–138.
- [92] ———, *Inertia-controlling methods for general quadratic programming*, SIAM Rev., **33** (1991), pp. 1–36.
- [93] ———, *Some theoretical properties of an augmented Lagrangian merit function*, in Advances in Optimization and Parallel Computing, P.M. Pardalos, ed., North Holland, North Holland, 1992, pp. 101–128.
- [94] P.E. GILL, W. MURRAY, AND M.H. WRIGHT, *Practical Optimization*, Academic Press, London and New York, 1981.
- [95] P.E. GILL AND D.P. ROBINSON, *A primal-dual augmented Lagrangian*, Computational Optimization and Applications (2010), pp. 1–25.
<http://dx.doi.org/10.1007/s10589-010-9339-1>.
- [96] P.E. GILL AND E. WONG, *Methods for convex and general quadratic programming*, Numerical Analysis Report 11-1, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2011.
- [97] ———, *A regularized method for convex and general quadratic programming*, Numerical Analysis Report 10-2, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2010.

- [98] D. GOLDFARB, *A family of variable metric methods derived by variational means*, Math. Comp., **24** (1970), pp. 23–26.
- [99] ———, *Curvilinear path steplength algorithms for minimization which use directions of negative curvature*, Math. Program., **18** (1980), pp. 31–40.
- [100] D. GOLDFARB AND A. IDNANI, *A numerically stable dual method for solving strictly convex quadratic programs*, Math. Programming, **27** (1983), pp. 1–33.
- [101] N.I.M. GOULD, *On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem*, Math. Program., **32** (1985), pp. 90–99.
- [102] ———, *On the accurate determination of search directions for simple differentiable penalty functions*, IMA J. Numer. Anal., **6** (1986), pp. 357–372.
- [103] ———, *An algorithm for large-scale quadratic programming*, IMA J. Numer. Anal., **11** (1991), pp. 299–324.
- [104] N.I.M. GOULD, D. ORBAN, AND PH.L. TOINT, *GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization*, ACM Trans. Math. Software, **29** (2003), pp. 353–372.
- [105] N.I.M. GOULD AND D.P. ROBINSON, *A second derivative SQP method with imposed descent*, Numerical Analysis Report 08/09, Computational Laboratory, University of Oxford, Oxford, UK, 2008.
- [106] ———, *A second derivative SQP method: Global convergence*, SIAM J. Optim., **20** (2010), pp. 2023–2048.
- [107] ———, *A second derivative SQP method: Local convergence and practical issues*, SIAM J. Optim., **20** (2010), pp. 2049–2079.
- [108] N.I.M. GOULD, J.A. SCOTT, AND Y. HU, *A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations*, ACM Trans. Math. Software, **33** (2007), pp. Art. 10, 32.
- [109] N.I.M. GOULD AND PH.L. TOINT, *An iterative working-set method for large-scale nonconvex quadratic programming*, Appl. Numer. Math., **43** (2002), pp. 109–128. 19th Dundee Biennial Conference on Numerical Analysis (2001).
- [110] ———, *Numerical methods for large-scale non-convex quadratic programming*, in Trends in industrial and applied mathematics (Amritsar, 2001), Vol. **72** of Appl. Optim., Kluwer Acad. Publ., Dordrecht, 2002, pp. 149–179.
- [111] J.-P. GOUX AND S. LEYFFER, *Solving large MINLPs on computational grids*, Optim. Eng., **3** (2002), pp. 327–346. Special issue on mixed-integer programming and its applications to engineering.
- [112] J. GREENSTADT, *On the relative efficiencies of gradient methods*, Math. Comp., **21** (1967), pp. 360–367.
- [113] L. GRIPPO, F. LAMPARIELLO, AND S. LUCIDI, *Newton-type algorithms with non-monotone line search for large-scale unconstrained optimization*, in System modelling and optimization (Tokyo, 1987), Vol. **113** of Lecture Notes in Control and Inform. Sci., Springer, Berlin, 1988, pp. 187–196.
- [114] ———, *A truncated Newton method with nonmonotone line search for unconstrained optimization*, J. Optim. Theory Appl., **60** (1989), pp. 401–419.
- [115] ———, *A class of nonmonotone stabilization methods in unconstrained optimization*, Numer. Math., **59** (1991), pp. 779–805.
- [116] N.-Z. GU AND J.-T. MO, *Incorporating nonmonotone strategies into the trust region method for unconstrained optimization*, Comput. Math. Appl., **55** (2008), pp. 2158–2172.
- [117] W.W. HAGER, *Stabilized sequential quadratic programming*, Comput. Optim. Appl., **12** (1999), pp. 253–273. Computational optimization—a tribute to Olvi Mangasarian, Part I.
- [118] S.P. HAN, *Superlinearly convergent variable metric algorithms for general nonlinear programming problems*, Math. Programming, **11** (1976/77), pp. 263–282.
- [119] ———, *A globally convergent method for nonlinear programming*, J. Optim. Theory Appl., **22** (1977), pp. 297–309.

- [120] S.P. HAN AND O.L. MANGASARIAN, *Exact penalty functions in nonlinear programming*, Math. Programming, **17** (1979), pp. 251–269.
- [121] J. HERSKOVITS, *A two-stage feasible directions algorithm for nonlinear constrained optimization*, Math. Programming, **36** (1986), pp. 19–38.
- [122] M.R. HESTENES, *Multiplier and gradient methods*, J. Optim. Theory Appl., **4** (1969), pp. 303–320.
- [123] H.M. HUYNH, *A Large-Scale Quadratic Programming Solver Based on Block-LU Updates of the KKT System*, PhD thesis, Program in Scientific Computing and Computational Mathematics, Stanford University, Stanford, CA, 2008.
- [124] M.M. KOSTREVA AND X. CHEN, *A superlinearly convergent method of feasible directions*, Appl. Math. Comput., **116** (2000), pp. 231–244.
- [125] ———, *Asymptotic rates of convergence of SQP-type methods of feasible directions*, in Optimization methods and applications, Vol. **52** of Appl. Optim., Kluwer Acad. Publ., Dordrecht, 2001, pp. 247–265.
- [126] J. KROYAN, *Trust-Search Algorithms for Unconstrained Optimization*, PhD thesis, Department of Mathematics, University of California, San Diego, February 2004.
- [127] C.T. LAWRENCE AND A.L. TITS, *A computationally efficient feasible sequential quadratic programming algorithm*, SIAM J. Optim., **11** (2001), pp. 1092–1118 (electronic).
- [128] S. LEYFFER, *Integrating SQP and branch-and-bound for mixed integer nonlinear programming*, Comput. Optim. Appl., **18** (2001), pp. 295–309.
- [129] D.C. LIU AND J. NOCEDAL, *On the limited memory BFGS method for large scale optimization*, Math. Program., **45** (1989), pp. 503–528.
- [130] X.-W. LIU AND Y.-X. YUAN, *A robust algorithm for optimization with general equality and inequality constraints*, SIAM J. Sci. Comput., **22** (2000), pp. 517–534 (electronic).
- [131] C.M. MAES, *A Regularized Active-Set Method for Sparse Convex Quadratic Programming*, PhD thesis, Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA, August 2010.
- [132] A. MAJTHAY, *Optimality conditions for quadratic programming*, Math. Programming, **1** (1971), pp. 359–365.
- [133] O.L. MANGASARIAN AND S. FROMOVITZ, *The Fritz John necessary optimality conditions in the presence of equality and inequality constraints*, J. Math. Anal. Appl., **17** (1967), pp. 37–47.
- [134] N. MARATOS, *Exact Penalty Function Algorithms for Finite-Dimensional and Control Optimization Problems*, PhD thesis, Department of Computing and Control, University of London, 1978.
- [135] J. MO, K. ZHANG, AND Z. WEI, *A variant of SQP method for inequality constrained optimization and its global convergence*, J. Comput. Appl. Math., **197** (2006), pp. 270–281.
- [136] J.L. MORALES, *A numerical study of limited memory BFGS methods*, Appl. Math. Lett., **15** (2002), pp. 481–487.
- [137] J.L. MORALES, J. NOCEDAL, AND Y. WU, *A sequential quadratic programming algorithm with an additional equality constrained phase*, Tech. Rep. OTC-05, Northwestern University, 2008.
- [138] J.J. MORÉ AND D.C. SORENSEN, *On the use of directions of negative curvature in a modified Newton method*, Math. Program., **16** (1979), pp. 1–20.
- [139] ———, *Newton's method*, in Studies in Mathematics, Volume **24**. MAA Studies in Numerical Analysis, G.H. Golub, ed., Math. Assoc. America, Washington, DC, 1984, pp. 29–82.
- [140] J.J. MORÉ AND D. J. THUENTE, *Line search algorithms with guaranteed sufficient decrease*, ACM Trans. Math. Software, **20** (1994), pp. 286–307.
- [141] W. MURRAY, *An algorithm for constrained minimization*, in Optimization (Sympos., Univ. Keele, Keele, 1968), Academic Press, London, 1969, pp. 247–258.

- [142] W. MURRAY AND F.J. PRIETO, *A sequential quadratic programming algorithm using an incomplete solution of the subproblem*, SIAM J. Optim., **5** (1995), pp. 590–640.
- [143] J. NOCEDAL AND S.J. WRIGHT, *Numerical Optimization*, Springer-Verlag, New York, 1999.
- [144] A. OLIVARES, J.M. MOGUERZA, AND F.J. PRIETO, *Nonconvex optimization using negative curvature within a modified linesearch*, European J. Oper. Res., **189** (2008), pp. 706–722.
- [145] J.M. ORTEGA AND W.C. RHEINBOLDT, *Iterative solution of nonlinear equations in several variables*, Academic Press, New York, 1970.
- [146] ———, *Iterative solution of nonlinear equations in several variables*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. Reprint of the 1970 original.
- [147] E.R. PANIER AND A.L. TITS, *A superlinearly convergent feasible method for the solution of inequality constrained optimization problems*, SIAM J. Control Optim., **25** (1987), pp. 934–950.
- [148] ———, *On combining feasibility, descent and superlinear convergence in inequality constrained optimization*, Math. Programming, **59** (1993), pp. 261–276.
- [149] P.M. PARDALOS AND G. SCHNITGER, *Checking local optimality in constrained quadratic programming is NP-hard*, Oper. Res. Lett., **7** (1988), pp. 33–35.
- [150] P.M. PARDALOS AND S.A. VAVASIS, *Quadratic programming with one negative eigenvalue is NP-hard*, J. Global Optim., **1** (1991), pp. 15–22.
- [151] M.J.D. POWELL, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., London and New York, 1969, Academic Press, pp. 283–298.
- [152] ———, *The convergence of variable metric methods for nonlinearly constrained optimization calculations*, in Nonlinear Programming, 3 (Proc. Sympos., Special Interest Group Math. Programming, Univ. Wisconsin, Madison, Wis., 1977), Academic Press, New York, 1978, pp. 27–63.
- [153] ———, *A fast algorithm for nonlinearly constrained optimization calculations*, in Numerical Analysis, Dundee 1977, G.A. Watson, ed., no. 630 in Lecture Notes in Mathematics, Heidelberg, Berlin, New York, 1978, Springer Verlag, pp. 144–157.
- [154] ———, *On the quadratic programming algorithm of Goldfarb and Idnani*, Math. Programming Stud., (1985), pp. 46–61.
- [155] D.P. ROBINSON, *Primal-Dual Methods for Nonlinear Optimization*, PhD thesis, Department of Mathematics, University of California, San Diego, September 2007.
- [156] S.M. ROBINSON, *A quadratically-convergent algorithm for general nonlinear programming problems*, Math. Program., **3** (1972), pp. 145–156.
- [157] ———, *Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms*, Math. Program., **7** (1974), pp. 1–16.
- [158] R.T. ROCKAFELLAR, *Augmented Lagrange multiplier functions and duality in nonconvex programming*, SIAM J. Control Optim., **12** (1974), pp. 268–285.
- [159] O. SCHENK AND K. GÄRTNER, *Solving unsymmetric sparse systems of linear equations with PARDISO*, in Computational science—ICCS 2002, Part II (Amsterdam), Vol. **2330** of Lecture Notes in Comput. Sci., Springer, Berlin, 2002, pp. 355–363.
- [160] K. SCHITTkowski, *The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function. I. Convergence analysis*, Numer. Math., **38** (1981/82), pp. 83–114.
- [161] ———, *The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function. II. An efficient implementation with linear least squares subproblems*, Numer. Math., **38** (1981/82), pp. 115–127.

- [162] ———, *On the convergence of a sequential quadratic programming method with an augmented Lagrangian line search function*, Math. Operationsforsch. Statist. Ser. Optim., **14** (1983), pp. 197–216.
- [163] R.B. SCHNABEL AND E. ESKOW, *A new modified Cholesky factorization*, SIAM J. Sci. and Statist. Comput., **11** (1990), pp. 1136–1158.
- [164] D.F. SHANNO, *Conditioning of quasi-Newton methods for function minimization*, Math. Comp., **24** (1970), pp. 647–656.
- [165] R.A. TAPIA, *A stable approach to Newton's method for general mathematical programming problems in \mathbb{R}^n* , J. Optim. Theory Appl., **14** (1974), pp. 453–476.
- [166] ———, *Diagonalized multiplier methods and quasi-Newton methods for constrained optimization*, J. Optim. Theory Appl., **22** (1977), pp. 135–194.
- [167] PH. L. TOINT, *An assessment of nonmonotone linesearch techniques for unconstrained optimization*, SIAM J. Sci. Comput., **17** (1996), pp. 725–739.
- [168] S. ULRICH, *On the superlinear local convergence of a filter-SQP method*, Math. Program., **100** (2004), pp. 217–245.
- [169] G. VAN DER HOEK, *Asymptotic properties of reduction methods applying linearly equality constrained reduced problems*, Math. Program., **16** (1982), pp. 162–189.
- [170] A. WÄCHTER AND L.T. BIEGLER, *Line search filter methods for nonlinear programming: local convergence*, SIAM J. Optim., **16** (2005), pp. 32–48 (electronic).
- [171] ———, *Line search filter methods for nonlinear programming: motivation and global convergence*, SIAM J. Optim., **16** (2005), pp. 1–31 (electronic).
- [172] R.B. WILSON, *A Simplicial Method for Convex Programming*, PhD thesis, Harvard University, 1963.
- [173] S.J. WRIGHT, *Superlinear convergence of a stabilized SQP method to a degenerate solution*, Comput. Optim. Appl., **11** (1998), pp. 253–275.
- [174] ———, *Modifying SQP for degenerate problems*, SIAM J. Optim., **13** (2002), pp. 470–497.
- [175] ———, *An algorithm for degenerate nonlinear programming with rapid local convergence*, SIAM J. Optim., **15** (2005), pp. 673–696.
- [176] Y.-X. YUAN, *Conditions for convergence of trust region algorithms for nonsmooth optimization*, Math. Programming, **31** (1985), pp. 220–228.
- [177] ———, *On the superlinear convergence of a trust region algorithm for nonsmooth optimization*, Math. Programming, **31** (1985), pp. 269–285.
- [178] ———, *On the convergence of a new trust region algorithm*, Numer. Math., **70** (1995), pp. 515–539.
- [179] W.I. ZANGWILL, *Non-linear programming via penalty functions*, Management Sci., **13** (1967), pp. 344–358.
- [180] H. ZHANG AND W.W. HAGER, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM J. Optim., **14** (2004), pp. 1043–1056 (electronic).

USING INTERIOR-POINT METHODS WITHIN AN OUTER APPROXIMATION FRAMEWORK FOR MIXED INTEGER NONLINEAR PROGRAMMING

HANDE Y. BENSON*

Abstract. Interior-point methods for nonlinear programming have been demonstrated to be quite efficient, especially for large scale problems, and, as such, they are ideal candidates for solving the nonlinear subproblems that arise in the solution of mixed-integer nonlinear programming problems via outer approximation. However, traditionally, infeasible primal-dual interior-point methods have had two main perceived deficiencies: (1) lack of infeasibility detection capabilities, and (2) poor performance after a warmstart. In this paper, we propose the exact primal-dual penalty approach as a means to overcome these deficiencies. The generality of this approach to handle any change to the problem makes it suitable for the outer approximation framework, where each nonlinear subproblem can differ from the others in the sequence in a variety of ways. Additionally, we examine cases where the nonlinear subproblems take on special forms, namely those of second-order cone programming problems and semidefinite programming problems. Encouraging numerical results are provided.

Key words. interior-point methods, nonlinear programming, integer programming.

AMS(MOS) subject classifications. 90C51, 90C11, 90C30, 90C25.

1. Introduction. The optimization problem considered in this paper is the Mixed Integer Nonlinear Programming (MINLP) problem of the form

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \\ \text{s.t.} \quad & h(x,y) \geq 0 \\ & A_x x \leq b_x \\ & A_y y \leq b_y \\ & y \in \mathcal{Z}^p, \end{aligned} \tag{1.1}$$

where $x \in \mathcal{R}^n$, $f : \mathcal{R}^{n+p} \rightarrow \mathcal{R}$ and $h : \mathcal{R}^{n+p} \rightarrow \mathcal{R}^{m_n}$ are twice continuously differentiable, $A_x \in \mathcal{R}^{m_x \times n}$, $A_y \in \mathcal{R}^{m_y \times p}$, $b_x \in \mathcal{R}^{m_x}$, $b_y \in \mathcal{R}^{m_y}$, and the linear constraints define polyhedral sets \mathcal{X} and \mathcal{Y} , which we assume to be bounded. When $p = 0$, we have the standard nonlinear programming problem (NLP), and when $n = 0$, we have an integer nonlinear programming problem. The constraints $h(x,y) \geq 0$ are nonlinear, and they can take special forms such as second-order cone constraints:

$$\hat{z} - \left\| \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} \right\|_2 \geq 0,$$

where \hat{z} is a scalar equal to one of the elements of x or y , \hat{x} and \hat{y} are vectors consisting of some or all elements of x and y , respectively, and $\|\cdot\|_2$ denotes

*Department of Decision Sciences, Bennett S. LeBow School of Business, Drexel University, Philadelphia, PA 19104 (benson@drexel.edu). Research supported by NSF grant CCF-0725692.

the Euclidean norm. These special forms include nonlinear formulations of semidefinite constraints as well, as shown in [10], [13], and [14].

The existing algorithms for solving a problem of the form (1.1) employ a two-level approach. In Branch-and-Bound ([28], [25]), the outer level successively partitions the feasible region of (1.1) by introducing or modifying bounds on y , while the inner level solves the continuous subproblems obtained by relaxing the integer constraints. In Outer Approximation ([18],[31]), the outer level solves a mixed-integer linear programming problem derived by the linearization of the objective and constraint functions at the solutions of the inner problem which is obtained by fixing the values of y in the original MINLP. Generalized Benders Decomposition [22] similarly alternates between the solution of a mixed-integer linear programming problem, but in the dual sense, and a continuous inner problem after fixing y . Other approaches, such as cutting-plane algorithms [3] exist for special forms of (1.1) including second-order cone programming problems. Software implementing these methods include SBB [21], MINLP [26], BARON [32], DICOPT [36], ALPHAEC [38], FILMINT [1], and BONMIN [11].

Regardless of the approach chosen to solve (1.1), a sequence of continuous optimization problems need to be solved, and the solution of these problems can account for a significant portion of the total runtime. Therefore, the solution algorithm employed to solve these problems must be efficient. A major source of this efficiency stems from the ability to reuse information obtained from solving related problems, or *warmstarting*. The solution algorithm must also be provably convergent in a sense that guarantees to find the global optimum for the continuous problem when such a solution exists and to issue a certificate of infeasibility when it does not. Failing on any single continuous relaxation will mean the failure of the overall algorithm.

In this paper, we will examine the use of an interior-point method as the inner level solution algorithm. Lack of warmstart and infeasibility detection capabilities have long been the main perceived difficulties of interior-point methods. Restarting from the solution of a previous problem may lead the algorithm to encounter numerical problems or even to stall, since the complementarity conditions lead to some nonnegative variables to be on the boundary at the given solution. For an infeasible interior-point method, it may be advantageous to start and remain infeasible throughout the solution process, and therefore, issuing a certificate of infeasibility for the problem in general is rather difficult. Additionally, a primal-dual interior point method seeks the analytic centers of the faces of optimal primal and dual solutions. Constraint qualifications that confirm the existence and finiteness of both primal and dual solutions are required to guarantee convergence. In fact, only one of the MINLP codes mentioned above, [11], uses a pure interior-point method, that is implemented in IPOPT [37], to solve the inner problem. Nevertheless, numerical studies such as

[9], [8], and [29] demonstrate that interior-point solvers such as IPOPT [37], LOQO [34], and KNITRO [30] are highly efficient and are the only solvers capable of handling very large scale NLPs. Therefore, it is important to resolve difficulties associated with warmstarting and infeasibility detection to implement an efficient and robust MINLP solver using interior-point methods.

In [5], we analyzed the use of an interior-point method within a branch-and-bound framework. We showed that the changing bounds would guarantee that the algorithm would stall when warmstarting, and that even with a coldstart, fixed variables and infeasible problems would cause the algorithm to fail. As a remedy, we proposed a primal-dual penalty approach, which was able to greatly improve efficiency, handle fixed variables, and correctly identify all infeasible subproblems in numerical testing.

In this paper, we turn our attention to interior-point methods within the Outer Approximation framework. Similar challenges arise in this framework, as well. One key difference is that we will limit ourselves to MINLPs with convex continuous relaxations, that is, cases where f is convex and h are concave for (1.1). This is required for the underlying theory of the Outer Approximation framework, and, while it is a limitation, it will also give us the chance to explore certain special classes of convex problems, such as second-order cone programming problems (SOCPs) and semidefinite programming problems (SDPs), that arise in the continuous relaxations.

The outline of the paper is as follows: We start with a brief description of the Outer Approximation framework in Section 2. In Section 3, we introduce an infeasible interior-point method and analyze its challenges within a MINLP algorithm. To address these challenges, we propose the exact primal-dual penalty method. In Section 4, we turn our attention to the performance of our algorithm on certain special classes of problems, such as SOCPs and SDPs. We present implementation details of our approach and favorable numerical results on problems from literature in Section 5.

2. Outer approximation. The Outer Approximation (OA) algorithm solves an alternating sequence of NLPs and mixed-integer linear programming problems (MILPs) to solve (1.1). For each $y^k \in \mathcal{Y} \cap \mathcal{Z}^p$, the NLP to be solved is obtained from (1.1) by fixing $y = y^k$:

$$\begin{aligned} \min_x \quad & f(x, y^k) \\ \text{s.t.} \quad & h(x, y^k) \geq 0 \\ & A_x x \leq b_x. \end{aligned} \tag{2.1}$$

(2.1) may or may not have a feasible solution. As such, we let x^k denote the solution if one exists and the minimizer of infeasibility otherwise. We define $\mathcal{F}(\hat{\mathcal{Y}})$ as the set of all pairs of (x^k, y^k) where x^k is an optimal solution of (2.1) and $\mathcal{I}(\hat{\mathcal{Y}})$ as the set of all pairs of (x^k, y^k) where (2.1) is infeasible for $y^k \in \hat{\mathcal{Y}}$. We also define the following MILP:

$$\begin{aligned}
& \min_{x, y, z} && z \\
\text{s.t. } & f(x^k, y^k) + \nabla f(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq z, && \forall (x^k, y^k) \in \mathcal{F}(\hat{\mathcal{Y}}) \\
& h(x^k, y^k) + \nabla h(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \geq 0, && \forall (x^k, y^k) \in \mathcal{F}(\hat{\mathcal{Y}}) \\
& h(x^k, y^k) + \nabla h(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \geq 0, && \forall (x^k, y^k) \in \mathcal{I}(\hat{\mathcal{Y}}) \\
& A_x x && \leq b_x \\
& A_y y && \leq b_y \\
& y && \in \mathcal{Z}^p,
\end{aligned} \tag{2.2}$$

where $z \in \mathcal{R}$ is a dummy variable.

Assuming that f is convex and h are concave, (1.1) is equivalent to (2.2) for $\hat{\mathcal{Y}} = \mathcal{Y}$, as shown in [18], [20], and [11]. Of course, solving (2.2) for $\hat{\mathcal{Y}} = \mathcal{Y}$ requires the solution of (2.1) for every $y^k \in \mathcal{Y} \cap \mathcal{Z}^p$, which constitutes the worst-case scenario. Instead, at each iteration, we solve (2.2) with $\hat{\mathcal{Y}} \subseteq \mathcal{Y}$. Starting with $y^0 \in \mathcal{Y} \cap \mathcal{Z}^p$, we let $\hat{\mathcal{Y}} = \{\}$. Then, at each iteration $k = 0, \dots, M$, we solve (2.1) with y^k to obtain x^k , let $\hat{\mathcal{Y}} = \hat{\mathcal{Y}} \cup \{y^k\}$, and solve (2.2). The solution gives y^{k+1} , and we repeat the process. Throughout the iterations, we keep track of an upper bound on the optimal objective function value of (2.2). Letting the upper bound start at ∞ , we update it with the optimal objective function value of (2.1) whenever a solution exists. If this value is not less than the current upper bound, then we stop the algorithm and declare that the pair (x^k, y^k) which gave the current upper bound is the optimal solution to (1.1).

3. Interior-point methods. The OA approach described above requires the repeated solves of NLPs obtained by fixing the values of the integer variables y in (1.1). At iteration k of the OA algorithm, (2.1) is solved for a different value of y^k . For each value of y^k , therefore, we can expect changes to both the objective function and the constraints of (2.1). Depending on the implementation, these changes could even be reflected in the problem structure, including the number of constraints and the nonzero structures of the Jacobian and the Hessian. To solve (2.1), we use an interior-point method, for which we now provide an overview. A more detailed explanation can be found in [35].

For ease of notation, let us rewrite (2.1) as follows:

$$\begin{aligned}
& \min_x && f(x, y^k) \\
\text{s.t. } & g(x, y^k) \geq 0,
\end{aligned} \tag{3.1}$$

where

$$g(x, y^k) = \begin{bmatrix} h(x, y^k) \\ b_x - A_x x \end{bmatrix}.$$

We start by adding the nonnegative slacks $w \in \mathcal{R}^m$ to the inequality constraints in (3.1).

$$\begin{aligned} \min_{x,w} \quad & f(x, y^k) \\ \text{s.t.} \quad & g(x, y^k) - w = 0 \\ & w \geq 0. \end{aligned} \quad (3.2)$$

We incorporate the slacks in a logarithmic barrier term in the objective function and eliminate the nonnegativity constraints:

$$\begin{aligned} \min_{x,w} \quad & f(x, y^k) - \mu \sum_{i=1}^m \log w_i \\ \text{s.t.} \quad & g(x, y^k) - w = 0, \end{aligned} \quad (3.3)$$

where $\mu > 0$ is the barrier parameter.

Denoting the Lagrange multipliers by λ , the first order conditions for (3.3) are

$$\begin{aligned} \nabla f(x, y^k) - A(x, y^k)^T \lambda &= 0, \\ -\mu e + W \Lambda e &= 0, \\ g(x, y^k) - w &= 0, \end{aligned} \quad (3.4)$$

where e is the vector of all ones of appropriate dimension, $A(x, y^k)$ is the transpose of the Jacobian of the constraints, and W and Λ are diagonal matrices with entries from w and λ , respectively.

Newton's method is employed to iterate to a point satisfying (3.4). Letting

$$\begin{aligned} H(x, y^k, \lambda) &= \nabla^2 f(x, y^k) - \sum_{i=1}^m \lambda_i \nabla^2 g_i(x, y^k), \\ \sigma &= \nabla f(x, y^k) - A(x, y^k)^T \lambda, \\ \gamma &= \mu W^{-1} e - \lambda, \\ \rho &= w - g(x, y^k), \end{aligned} \quad (3.5)$$

the directions given by Newton's method are found by solving the *KKT system*:

$$\begin{bmatrix} -W^{-1}\Lambda & -I \\ -I & A \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -\gamma \\ \sigma \\ \rho \end{bmatrix}. \quad (3.6)$$

Note that we have omitted the use of function arguments for ease of display.

Letting

$$E = W \Lambda^{-1}$$

we can eliminate the slacks to obtain the *reduced KKT system*:

$$\begin{bmatrix} -H & A^T \\ A & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \sigma \\ \rho + E\gamma \end{bmatrix}. \quad (3.7)$$

The reduced KKT system is solved by using the LDL^T form of Cholesky factorization, including exploitation of sparsity by reordering the columns in a symbolic Cholesky routine. As stated before, for each y^k , the sparsity structure of the matrix in (3.7) may change. Such changes are quite common, especially when y are binary. Fixing y_j^k to 0 may cause terms in the objective or the constraint functions to drop. A careful implementation of the underlying algorithm can take advantage of such changes if they bring about substantial reduction in size or complexity for certain subproblems, or use a general enough sparsity structure so that each subsequent nonlinear subproblem can be solved without additional sparsity structure setups or calls to the symbolic Cholesky routine.

Once the step directions Δx and $\Delta \lambda$ are obtained from (3.7), we can obtain the step directions for the slack variables from the following formula:

$$\Delta w = W\Lambda^{-1}(\mu W^{-1}e - \lambda - \Delta \lambda). \quad (3.8)$$

The algorithm then proceeds to a new estimate of the optimum by

$$\begin{aligned} x^{(l+1)} &= x^{(l)} + \alpha^{(l)} \Delta x^{(l)} \\ \lambda^{(l+1)} &= \lambda^{(l)} + \alpha^{(l)} \Delta \lambda^{(l)} \\ w^{(l+1)} &= w^{(l)} + \alpha^{(l)} \Delta w^{(l)}, \end{aligned} \quad (3.9)$$

where the superscripts denote the iteration number, $\alpha^{(l)}$ is chosen to ensure that the slacks $w^{(l+1)}$ and the dual variables $\lambda^{(l+1)}$ remain strictly positive and sufficient progress toward optimality and feasibility is attained. At each iteration, the value of the barrier parameter may also be updated as a function of $W^{(l+1)}\lambda^{(l+1)}$. Both the notion of sufficient progress and the exact formula for the barrier parameter update vary from one solver to another, but the general principle remains the same.

The algorithm concludes that it has reached an optimal solution when the primal infeasibility, the dual infeasibility, and the average complementarity are all less than a given tolerance level. For (3.1), we have that

$$\begin{aligned} \text{primal infeasibility} &= \|\rho\|_\infty \\ \text{dual infeasibility} &= \|\sigma\|_\infty \\ \text{average complementarity} &= \frac{w^T \lambda}{m}, \end{aligned}$$

where $\|\cdot\|_\infty$ denotes the infinity norm.

3.1. Challenges when using interior-point methods. An infeasible interior-point method, such as the one described above, has two main challenges within the OA framework: guaranteeing that a certificate of infeasibility will be issued when a solution does not exist and warmstarting.

An interior-point method such as the one described above is also known as an *infeasible* interior-point method. This terminology is used to indicate that the initial values for the primal variables x are not required to be feasible for the problem. In fact, the iterates are not even required to be feasible until optimality is also attained. Therefore, an infeasible interior-point method can potentially iterate forever when attempting to solve an infeasible problem. In practice, the solver will stop after reaching a preset iteration limit, and the result will be inconclusive. This leads to a failure in the overall algorithm, as we cannot produce a certificate of optimality or infeasibility at an iteration. Therefore, an infeasibility detection scheme is required. This scheme could be a “Phase I” approach where the interior-point method is first employed to solve the problem of minimizing constraint violations. If a feasible solution is found, the algorithm proceeds toward the optimum from there. Otherwise, a certificate of infeasibility is issued. While detecting infeasibility early in the solution process, this approach could significantly increase the solution time when an optimal solution exists, since it essentially requires the solution of two problems. Another possibility is to use the so-called “elastic mode,” where the algorithm starts solving (2.1), but switches to minimizing the infeasibility only after certain trigger conditions are observed. Therefore, when an optimal solution to the original problem exists, it can be found within a single solve, and if the problem is infeasible, trigger conditions that switch over to the feasibility problem early enough can keep the number of iterations reasonable for issuing a certificate of infeasibility. However, in the case of solving an NLP using the interior-point method described above, defining such trigger conditions could be a challenge. A third possibility is to use a one-shot approach, where a reformulation of (2.1) is solved and the solution of this reformulation gives the optimal solution or a certificate of infeasibility for the original problem. An example is self-dual embedding, which is well-developed for second-order cone and semidefinite programming problems. Version for convex NLPs ([39],[27]) also exist.

Even if an optimal solution exists, the interior-point method described above may not be guaranteed to find it. Certain constraint qualifications required for standard convergence proofs, such as the Mangasarian-Fromowitz Constraint Qualification (MFCQ), may not be satisfied. Therefore, it is important to use an approach that is provably convergent under mild assumptions. Penalty methods, which reformulate the problem (2.1) and use interior-point methods to solve the resulting problem, are such approaches. [2], [6], [33], and [24] all use penalty methods, and the algorithms proposed in these papers make few assumptions, including differentiabil-

ity and boundedness of the iterates, without requiring strong constraint qualifications.

Warmstarting is the use of information obtained during the solution of a problem to solve the subsequent, closely-related problems. For the case of MINLP, warmstarting will refer specifically to setting the initial solution (including primal, dual, and slack variables) of an NLP of the form (2.1) to the optimal solution of the previous one solved within the OA framework. Because of the complementarity conditions, at the optimal solution, some of the nonnegative slack and dual variables are equal to 0, but starting the next problem from these values may cause the algorithm to stall. The following example illustrates exactly what can go wrong:

$$\begin{array}{ll} \min_{x,y} & (x - 0.25)^2 + y \\ \text{s.t.} & -60x^3 \geq -y \\ & x \geq 0 \\ & y \in \{0, 1\}. \end{array}$$

The reduced KKT system for this problem is:

$$\begin{bmatrix} -2 - 360x\lambda_1 & -180x^2 & 1 \\ -180x^2 & \frac{w_1}{\lambda_1} & 0 \\ 1 & 0 & \frac{w_2}{\lambda_2} \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda_1 \\ \Delta \lambda_2 \end{pmatrix} = \begin{pmatrix} 2x - 0.5 + 180x^2\lambda_1 \\ \frac{\mu}{\lambda_1} - (y - 60x^3) \\ \frac{\mu}{\lambda_2} - x \end{pmatrix}.$$

Let $y = 1$ for the first subproblem. Then, $x^* = 0.25$, $w^* = (0.062, 0.25)$, and $\lambda^* = (0, 0)$. Let $y = 0$ for the next subproblem. In the first iteration, $\Delta x = 0$, $\Delta \lambda_2 = 0$, $\Delta w_2 = 0$, and $\Delta \lambda_1 > 0$ but very close to 0. Then, $\Delta w_1 = \frac{\mu}{\lambda_1} - w_1 - \frac{w_1}{\lambda_1} \Delta \lambda_1 = -1$. The steplength is shortened to less than 0.062, and the algorithm becomes stuck at the old solution.

One possible remedy is to simply re-initialize the slack and dual variables away from 0. Doing so would modify both the diagonal matrix, D , in (3.7) and the right-hand side of (3.7), forcing the variables to move. While this seems like a simple remedy, there are two drawbacks to this approach. First, the initialization is rather arbitrary and may adversely affect the efficiency algorithm at the current node. Secondly, note that simply re-initializing some of the variables may result in negative step directions for other dual and slack variables. Since the interior-point method shortens the steplength to keep such variables strictly positive, the algorithm may still become stuck.

Traditional penalty approaches, which only provide primal relaxations, may still become stuck when used with a primal-dual interior-point method where the steplength α depends on the dual iterates as well. However, they have other desirable properties, including infeasibility detection capabilities and regularizations that automatically satisfy constraint qualifications. The exact primal-dual penalty method proposed in [7] for linear programming and in [8] for nonlinear programming is a remedy that has been

demonstrated to work for warmstarts. This method relaxes the nonnegativity constraint on the dual and slack variables and provides regularization for the matrix in the reduced KKT system (3.7). Thus, the optimal solution of one problem can be used without modification to provide a warmstart for another, the regularization ensures that the variables that need to move indeed make progress, and the algorithm does not become stuck due to the nonnegativity constraints. This approach was shown to work well for mixed-integer nonlinear programming within the branch-and-bound framework in [5]. Additional benefits include robustness due to regularization and infeasibility detection capabilities. Details of this approach for general nonlinear programming problems are given in [8], and we provide an overview here.

4. The exact primal-dual penalty approach. The primal-dual penalty problem corresponding to (3.1) has the form

$$\begin{aligned} \min_{x,w,\xi} \quad & f(x, y^k) + c^T \xi \\ \text{s.t.} \quad & g(x, y^k) - w = 0 \\ & -\xi \leq w \leq u \\ & \xi \geq 0, \end{aligned} \tag{4.1}$$

where $c \in \mathcal{R}^m$ are the primal penalty parameters, $u \in \mathcal{R}^m$ are the dual penalty parameters, and $\xi \in \mathcal{R}^m$ are the primal relaxation variables. This new form of the primal penalty problem differs from the classical approach presented in [19] in two crucial aspects: (1) The slacks, w , rather than the constraint functions and the bounds themselves are relaxed, and (2) upper bounds are also added to the slacks. Both of these changes are made specifically for warmstarting, as relaxing the slacks removes their nonnegativity constraints and allows for longer steps and the upper bounds serve to relax the dual variables in a similar manner. The dual problem associated with (4.1) can be expressed as follows:

$$\begin{aligned} \max_{\lambda, \psi} \quad & f(x, y^k) - \nabla f(x, y^k)^T x - (h(x, y^k) - A(x, y^k)x)^T \lambda - u^T \psi \\ \text{s.t.} \quad & \nabla f(x, y^k) - A(x, y^k)^T \lambda = 0 \\ & -\psi \leq \lambda \leq c - \psi \\ & \psi \geq 0, \end{aligned} \tag{4.2}$$

where $\psi \in \mathcal{R}^m$ are the dual relaxation variables. These relaxation variables are incorporated into the objective function using a penalty term with dual penalty parameters u . For further details of the primal and dual problems, as well as a proof of the exactness of the penalty approach, the reader is referred to [7] and [8].

We follow the development of earlier in Section 2 in order to present the algorithm to solve (4.1). The logarithmic barrier problem associated with (4.1) is

$$\begin{aligned}
& \min_{x, y^k, w, \xi} f(x, y^k) + c^T \xi \\
& \quad -\mu \sum_{i=1}^m \log(\xi_i) - \mu \sum_{i=1}^m \log(w_i + \xi_i) - \mu \sum_{i=1}^m \log(u_i - w_i) \\
\text{s.t. } & g(x, y^k) - w = 0,
\end{aligned} \tag{4.3}$$

where $\mu > 0$ is the barrier parameter. Letting (λ) once again denote the dual variables associated with the remaining constraints, the first-order conditions for the Lagrangian of (4.3) can be written as

$$\begin{aligned}
g(x, y^k) - w &= 0 \\
\nabla f(x, y^k) - A(x, y^k)^T \lambda &= 0 \\
\lambda - \mu(W + \Xi)^{-1}e + \mu(U - W)^{-1}e &= 0 \\
c - \mu\Xi^{-1}e - \mu(W + \Xi)^{-1}e &= 0
\end{aligned}$$

where Ξ and U are the diagonal matrices with the entries of ξ and u , respectively. Making the substitution

$$\psi = \mu(U - W)^{-1}e$$

we can rewrite the first-order conditions as

$$\begin{aligned}
g(x, y^k) - w &= 0 \\
\nabla f(x, y^k) - A(x, y^k)^T \lambda &= 0 \\
(W + \Xi)(\Lambda + \Psi)e &= \mu e \\
\Xi(C - \Lambda - \Psi)e &= \mu e \\
\Psi(U - W)e &= \mu e
\end{aligned} \tag{4.4}$$

where Ψ and C are the diagonal matrices with the entries of ψ and c , respectively. Note that the new variables ψ serve to relax the nonnegativity requirements on the dual variables λ , so we refer to them as the dual relaxation variables.

Applying Newton's Method to (4.4), and eliminating the step directions for w, ξ , and ψ , the reduced KKT system arising in the solution of the penalty problem (4.1) has the same form as (3.7) with

$$\begin{aligned}
E &= \left(((\Lambda + \Psi)^{-1}(W + \Xi) + \Xi(C - \Lambda - \Psi)^{-1})^{-1} + \Psi(U - W)^{-1} \right)^{-1} \\
\gamma &= \left((\Lambda + \Psi)^{-1}(W + \Xi) + \Xi(C - \Lambda - \Psi)^{-1} \right)^{-1} \\
&\quad (\mu(\Lambda + \Psi)^{-1}e - \mu(C - \Lambda - \Psi)^{-1}e - w) - (\mu(U - W)^{-1}e - \psi).
\end{aligned} \tag{4.5}$$

The steplength, $\alpha^{(k)}$, at each iteration k is chosen to ensure that

$$\begin{aligned}
w^{(k+1)} + \xi^{(k+1)} &> 0 \\
\lambda^{(k+1)} + \psi^{(k+1)} &> 0 \\
\xi^{(k+1)} &> 0 \\
\psi^{(k+1)} &> 0 \\
u - w^{(k+1)} &> 0 \\
c - \lambda^{(k+1)} - \psi^{(k+1)} &> 0
\end{aligned}$$

and sufficient progress toward optimality and feasibility is made. The barrier parameter, μ , may be updated at each iteration as a function of $(W + \Xi)(\Lambda + \Psi)e$, $\Xi(C - \Lambda - \Psi)e$, and $\Psi(U - W)e$.

There are several things to note about this approach. First, the sparsity structure of the reduced KKT matrix of the penalty problem is the same as the sparsity structure of (3.7). There are also no additional function evaluations or other time consuming computations required. This means that solving the penalty problem (4.1) instead of (3.1) does not require significant additional computational effort. Second, by modifying E , the relaxation/penalty scheme is said to regularize the reduced KKT matrix, providing numerical stability as well as aiding in warmstarting. Third, steplength control no longer relies on the dual and slack variables of the original problem, thereby allowing for longer steps in the initial iterations to ensure that the algorithm does not become stuck.

The primal-dual penalty approach presents an ideal remedy to the warmstarting issues of an interior-point method. For each NLP subproblem, we can use the optimal primal, dual, and slack variable values of the previous subproblem as the initial solution, and simply re-initialize the primal and dual relaxation variables in order to facilitate the original variables to move toward a new optimum. The penalty parameters need to be chosen large enough to admit the optimal solution of the subproblem, and warmstart information may be useful to determine appropriate values. They may also need to be updated during the course of the algorithm.

4.1. Setting and updating the penalty parameters. The most important aspect of setting the initial values of the penalty parameters is to ensure that they are sufficiently larger than those components of the current iterate for which they serve as upper bounds. We let the solution of one NLP subproblem be (x^*, w^*, λ^*) . The penalty parameters are set as follows:

$$\begin{aligned} u &= w^* + \kappa_w e \\ c &= \lambda^* + \psi^{(0)} + \kappa_\lambda e \end{aligned}$$

where

$$\begin{aligned} \kappa_w &= \max(g(x^*, y^*), w^*, 1.0) \\ \kappa_\lambda &= \max(\lambda^*, 1.0) \end{aligned}$$

The relaxation variables are initialized as

$$\begin{aligned} \xi^{(0)} &= \beta \kappa_w e \\ \psi^{(0)} &= \beta \kappa_\lambda e \end{aligned} \tag{4.6}$$

where β is a constant with a default value of 10^{-4} . These initializations are generally sufficient after a warmstart to start the penalty method without moving the iterates too far from the current point. Note that the relaxation is performed using a variable, so if a larger relaxation is needed, the variables, ξ and ψ , will move as necessary.

Since the initial values of the penalty parameters, u and c , may not be large enough to admit the optimal solution, we also need an updating scheme for these parameters. Given the relaxation, an optimal solution can always be found for (4.1), and one possible “static” updating scheme is to solve a problem to optimality and to increase the penalty parameters if their corresponding relaxation variables are not sufficiently close to zero. However, this may require multiple solves of a problem and substantially increase the number of iterations necessary to find the optimal solution. Instead, we can use a “dynamic” updating scheme, where the penalty parameters are checked at the end of each iteration and updated. For $i = 1, \dots, m + m_x$, if $w_i^{(k+1)} > 0.9u_i^{(k)}$, then $u_i^{(k+1)} = 10u_i^{(k)}$. Similarly, if $\lambda_i^{(k+1)} + \psi_i^{(k)} > 0.9c_i^{(k)}$, then $c_i^{(k+1)} = 10c_i^{(k)}$.

4.2. Infeasibility detection. In the preceding discussion, we established what can go wrong when warmstarting an interior-point method and proposed the exact primal-dual penalty approach as a remedy. Another concern for improving the inner level algorithm within our framework was the efficient identification of infeasible NLP subproblems. The primal-dual penalty method described as a remedy for warmstarting can also aid in infeasibility identification. Since all of the slack variables are relaxed, the penalty problem (4.1) always possesses a feasible solution. In addition, the upper bounds on the slack variables guarantee that an optimal solution to (4.1) always exists. Therefore, a provably convergent NLP algorithm is guaranteed to find an optimal solution to (4.1). If this solution has the property that $\xi_i \rightarrow a$ for at least one $i = 1, \dots, m + m_x$ for some scalar $a > 0$ as $c_i \rightarrow \infty$, then the original problem is infeasible.

It is impractical to allow a penalty parameter to become infinite. However, a practical implementation can be easily devised by simply dropping the original objective function and minimizing only the penalty term, which is equivalent to letting all the penalty parameters become infinite. Therefore, a feasibility restoration phase similar to the “elastic mode” of SNOPT [23] can be used, in that the problem

$$\begin{aligned} \min_{x, w, \xi} \quad & c^T \xi \\ \text{s.t.} \quad & g(x, y^k) - w = 0 \\ & -\xi \leq w \leq u \\ & \xi \geq 0, \end{aligned} \tag{4.7}$$

is solved in order to minimize infeasibility. It differs from SNOPT’s version in that the slack variables are still bounded above by the dual penalty parameters. Since these parameters get updated whenever necessary, we can always find a feasible solution to (4.7). If the optimal objective function value is nonzero (numerically, greater than the infeasibility tolerance), a certificate of infeasibility can be issued.

While a feasibility problem can be defined for the original NLP subproblem (2.1) as well, a trigger condition for switching into the “elastic

mode" for solving it is not easy to define within the context of the interior-point method of Section 3. However, the exact primal-dual penalty approach can simply track the number of dynamic updates made to the penalty parameters and switch over to solving (4.7) after a finite number of such updates are performed. In our numerical testing, we have set this trigger to occur after three updates to any single penalty parameter.

Note that other infeasibility detection schemes based on penalty methods are available (see [16]) which would not require the solution of a separate feasibility problem. As their warmstarting capabilities are yet unknown, we will investigate such approaches in future work.

5. Special forms of convex NLPs. One class of problems that fits well into the OA framework is conic programming, specifically second-order cone programming and semidefinite programming. This class of problems is especially important in a variety of engineering applications and as relaxations of some NP-hard combinatorial problems. Much of the research has focused on problems that are otherwise linear, due in part to the abundance of strong theoretical results and the ease of extending established and implemented linear programming algorithms. However, as the models in each of these areas become more realistic and more complicated, many of the problems are expressed with nonlinearities in the objective function and/or the constraints. To handle such nonlinearities efficiently, one approach is to fit the problem into the NLP framework through reformulation or separation into a series of NLP subproblems. In addition, these problems can also have some discrete variables, and fitting them into an NLP framework allows for the use of the efficient mixed-integer nonlinear programming techniques for their solution.

In standard form, a mixed-integer nonlinear cone programming problem is given by

$$\begin{aligned} \min_{x,y} \quad & f(x, y) \\ \text{s.t.} \quad & h(x, y) \geq 0 \\ & x \in \mathcal{K} \\ & y \in \mathcal{Y}, \end{aligned} \tag{5.1}$$

where \mathcal{K} is a cone. The second-order, or Lorentz, cone is defined by

$$\mathcal{K} = \{(x_0, x_1) \in \mathcal{R}^n : x_0 \in \mathcal{R}, x_1 \in \mathcal{R}^{n-1}, \|x_1\|_2 \leq x_0\}, \tag{5.2}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. The semidefinite cone is defined by

$$\mathcal{K} = \{x \in \mathcal{R}^n : \text{mat}(x) \succeq 0\}, \tag{5.3}$$

where $\text{mat}(x) \in \mathcal{R}^{k \times k}$ with $n = k^2$ is the MATLAB-like notation for the columnwise definition of a matrix from the vector x , and $\succeq 0$ constrains

this matrix to be symmetric and positive semidefinite. Note that \mathcal{K} can also represent the intersection of finitely many such cones.

The primal-dual penalty method can be applied to this problem just as in (4.1). The cone constraint can be handled as

$$\begin{aligned} x + \xi &\in \mathcal{K} \\ u - x &\in \mathcal{K} \\ \xi &\in \mathcal{K}. \end{aligned} \tag{5.4}$$

For a second order cone, it is sufficient to pick $\xi = (\xi_0, 0)$, and for a semidefinite cone, we only need $\text{mat}(\xi)$ to be a diagonal matrix. As before, the objective function is also converted to

$$f(x, y) + c^T \xi.$$

Since both the second-order cone and the cone of positive semidefinite matrices are self-dual, the dual problem also involves a cone constraint, which is similarly relaxed and bounded.

For both second-order and semidefinite cones, the reformulation of the cone constraints to fit into the NLP framework have been extensively discussed in [10]. For second-order cones, an additional challenge is the nondifferentiability of the Euclidean norm in (5.2). In fact, if the optimal solution includes $x_1^* = 0$, it can cause numerical problems for convergence of the NLP algorithm and theoretical complications for the formulation of the subsequent MILP even if numerical convergence can be attained for the NLP. There are several ways around this issue: if a preprocessor is used and a nonzero lower bound for x_0 is available, then the so-called *ratio reformulation* (see [10]) can be used to rewrite the cone constraint of (5.2) as

$$\frac{x_1^T x_1}{x_0} \leq x_0, \quad x_0 \geq 0.$$

Similarly, if preprocessing can determine that $\|x_1\|_2$ and x_0 are bounded above by small values, then we can rewrite the cone constraint as

$$e^{(x_1^T x_1 - x_0^2)/2} \leq 1, \quad x_0 \geq 0.$$

Both of these formulations yield convex NLPs, but they are not general enough. In our implementation, we have used the constraint as given in (5.2), but a more thorough treatment using a subgradient approach is discussed in [17].

6. Numerical results. We implemented an OA framework and the interior-point method using the primal-dual penalty approach in the solver MILANO [4]. For comparison purposes, we also implemented the interior-point method outlined at the beginning of Section 2. The MILPs that arise

within the OA framework were solved using a branch-and-bound algorithm using interior-point methods to solve the LP relaxations. We tested both codes on 12 problems from the MINLPLib [15] test suite and 2 MINLPs with second-order cone constraints from [17]. The problems were chosen to have convex nonlinear relaxations, to be small for easy implementation in MATLAB, and to require more than one NLP subproblem in the solution process so that the effect of warmstarting could be measured. We included only two of the small problems from [17] because several of the remaining problems had artificial continuous variables and equality constraints and only had integer variables when converted to the form (2.1). Since MILANO is implemented for the MATLAB environment, we converted the problems from MINLPLib from the GAMS [12] format to MATLAB format.

The initial primal and dual solutions used when warmstarting are the optimal primal and dual solutions of the previous NLP subproblem. For coldstarts, we used any user-provided initial solutions, and where none were available, the primal variable was initialized to 0 and all nonnegative slack and dual variables were initialized to 1. Numerical experience in Table 1 indicates that using this solution can improve the performance of the algorithm. However, a better primal initial solution can be the optimal x values from the current MILP. In this case, we would need to use an approximation to the Lagrange multipliers, for example by approximately solving a QP model of the NLP subproblem. This will be part of our future work.

In Table 1, we present results highlighting the effect of the primal-dual penalty approach on the interior-point method. In our testing, we have the primal-dual penalty approach determine the subproblems to be solved, and the columns “WarmIters” and “ColdIters” provide the average number of iterations over those subproblems after a warmstart using the primal-dual penalty approach and a coldstart using the original form of the interior-point method, respectively. The column “%Impr” indicates the percentage improvement in the number of iterations. This number is not always positive, but the warmstart approach is never more than 17% worse than the coldstart approach. The worsening can be remedied in many cases using different initial values for the penalty parameters and the relaxation variables. In the remaining 30 of the 32 subproblems solved, the percentage improvement can range from 0 to 65%.

We also provide information on the infeasible problems identified by the penalty approach. Since the original formulation of the interior-point method has no mechanism with which to issue a certificate of infeasibility, the coldstart algorithm goes to an iteration limit of 500 for each infeasible subproblem, after making a significant computational effort. This means that for problems *alan*, *fuel*, *gbd*, and *synthes2*, the OA algorithm would fail after encountering an infeasible NLP subproblem.

TABLE 1

Comparison of the warmstarting primal-dual penalty approach to coldstarts on small problems from the MINLPLib test suite and two mixed-integer second-order cone programming problems from [17] (indicated by “*” in the table). “#” indicates the NLP subproblem being solved, WarmIters and ColdIters are the numbers of warmstart and coldstart iterations, respectively, and %Impr is the percentage of improvement in the number of iterations. (INF) indicates that a certificate of infeasibility was issued, and (IL) denotes that the algorithm reached its iteration limit.

NAME	<i>n</i>	<i>p</i>	<i>m + m_x</i>	#	WarmIters	ColdIters	%Impr
alan	4	4	6	1	(INF)	(IL)	—
				2	8	10	20.00
				3	10	13	23.07
				4	11	13	15.38
				5	9	13	30.77
				6	9	10	10.00
bsp5var2*	2	1	4	1	7	7	0.00
				2	6	7	14.29
ex1223a	3	4	9	1	10	13	23.07
				2	7	12	41.67
ex1223b	3	4	9	1	11	13	15.38
				2	12	15	20.00
				3	9	12	25.00
				4	9	11	18.18
				5	10	11	9.09
ex1223	7	4	13	1	13	12	-8.30
				2	13	15	13.33
				3	15	16	6.25
				4	12	15	20.00
				5	14	12	-16.67
fuel	12	3	15	1	(INF)	(IL)	—
				2	18	51	64.71
gbd	1	3	2	1	(INF)	(IL)	—
				2	7	9	22.22
gkocis	8	3	8	1	15	14	-7.14
				2	11	12	8.33
oaer	6	3	6	1	13	12	-8.33
				2	12	16	25.00
procsel	7	3	7	1	10	10	0.00
				2	9	12	25.00
				3	9	10	10.00
				4	10	10	0.00
st_el14	7	4	13	1	11	13	15.38
				2	12	15	20.00
				3	9	12	25.00
				4	9	11	18.18
				5	10	11	9.09
synthes1	3	3	5	1	15	14	-7.14
				2	12	12	0.00
				3	12	14	14.29
synthes2	6	5	12	1	(INF)	(IL)	—
				2	15	15	0.00
				3	9	14	35.71
				4	9	18	50.00
				5	10	17	41.18
test5*	1	4	3	1	9	9	0.00
				2	6	8	25.00

7. Conclusion. In this paper, we described the solution of a mixed-integer nonlinear programming problem using an interior-point method within the context of an outer approximation algorithm. We resolved the issues of warmstarting, infeasibility detection, and robustness for the interior-point method. In doing so, we used the exact primal-dual penalty method of [7] and [8]. The resulting algorithm was implemented using the interior-point code MILANO [4] and tested on a suite of MINLPs. The numerical testing yielded encouraging results.

As discussed, interior-point codes have been shown to be computationally superior to other approaches in studies such as [9] for large problems. Therefore, the proposed approach is especially attractive for large MINLPs, where an interior-point code may be the only means of obtaining a solution to each continuous relaxation in a reasonable amount of time. The use of the primal-dual penalty method further improves the robustness and the efficiency of this approach.

The next step in this study is to incorporate the proposed approach within a more efficient algorithm to handle the integer variables and introduce heuristics for generating feasible solutions quickly. Numerical results in [7] and [8] demonstrate the strong performance of the primal-dual penalty approach under a variety of problem modifications, including the addition of constraints and variables. Thus, we are optimistic that the performance improvements demonstrated in this paper will continue to be applicable when used within any integer programming framework.

Acknowledgements. The author wishes to thank Sven Leyffer and an anonymous referee for their helpful comments and suggestions.

REFERENCES

- [1] K. ABHISHEK, S. LEYFFER, AND J. LINDEROTH, *Filmint: An outer-approximation-based solver for nonlinear mixed integer programs*, Tech. Rep. Preprint ANL/MCS-P1374-0906, Argonne National Laboratory, Mathematics and Science Division, September 2006.
- [2] P. ARMAND, *A quasi-newton penalty barrier method for convex minimization problems*, Computational Optimization and Applications, **26** (2003), pp. 5–34.
- [3] A. ATAMTÜRK AND V. NARAYANAN, *Conic mixed-integer rounding cuts*, Research Report BCOL.06.03, IEOR, University of California-Berkeley, December 2006.
- [4] H. BENSON, *MILANO - a Matlab-based code for mixed-integer linear and nonlinear optimization*. <http://www.pages.drexel.edu/~hvb22/milano>.
- [5] ———, *Mixed-integer nonlinear programming using interior-point methods*, tech. rep., Submitted to *Optimization Methods and Software*, November 2007.
- [6] H. BENSON, A. SEN, AND D. SHANNO, *Convergence analysis of an interior-point method for nonconvex nonlinear programming*, tech. rep., Submitted to *Mathematical Programming Computation*, February 2009.
- [7] H. BENSON AND D. SHANNO, *An exact primal-dual penalty method approach to warmstarting interior-point methods for linear programming*, Computational Optimization and Applications, **38** (2007), pp. 371–399.

- [8] ———, *Interior-point methods for nonconvex nonlinear programming: Regularization and warmstarts*, Computational Optimization and Applications, **40** (2008), pp. 143–189.
- [9] H. BENSON, D. SHANNO, AND R. VANDERBEI, *Interior-point methods for nonconvex nonlinear programming: Filter methods and merit functions*, Computational Optimization and Applications, **23** (2002), pp. 257–272.
- [10] H. BENSON AND R. VANDERBEI, *Solving problems with semidefinite and related constraints using interior-point methods for nonlinear programming*, Mathematical Programming B, **95** (2003), pp. 279–302.
- [11] P. BONAMI, L. BIEGLER, A. CONN, G. CORNUEJOLS, I. GROSSMAN, C. LAIRD, J. LEE, A. LODI, F. MARGOT, N. SAWAYA, AND A. WAECHTER, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization, **5** (2008), pp. 186–204.
- [12] A. BROOKE, D. KENDRICK, AND A. MEERAUS, *GAMS: A User's Guide*, Scientific Press, 1988.
- [13] S. BURER, R. MONTEIRO, AND Y. ZHANG, *Solving semidefinite programs via nonlinear programming part I: Transformations and derivatives*, tech. rep., TR99-17, Dept. of Computational and Applied Mathematics, Rice University, Houston TX, 1999.
- [14] ———, *Solving semidefinite programs via nonlinear programming part II: Interior point methods for a subclass of SDPs*, tech. rep., TR99-17, Dept. of Computational and Applied Mathematics, Rice University, Houston TX, 1999.
- [15] M. BUSSIECK, A. DRUD, AND A. MEERAUS, *MINLPLib - a collection of test models for mixed-integer nonlinear programming*, INFORMS Journal on Computing, **15**(1) (2003), pp. 114–119.
- [16] R. H. BYRD, F. E. CURTIS, AND J. NOCEDAL, *Infeasibility detection and sqp methods for nonlinear optimization*, SIAM Journal on Optimization, **20** (2010), pp. 2281–2299.
- [17] S. DREWES, *Mixed integer second order cone programming*. PhD thesis. Technischen Universität Darmstadt, Munich, Germany., 2009.
- [18] M. DURAN AND I. GROSSMANN, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Mathematical Programming, **36** (1986), pp. 307–339.
- [19] R. FLETCHER, *Practical Methods of Optimization*, J. Wiley and Sons, Chichester, England, 1987.
- [20] R. FLETCHER AND S. LEYFFER, *Solving mixed integer nonlinear programs by outer approximation*, Mathematical Programming, **66** (1994), pp. 327–349.
- [21] GAMS, *GAMS-SBB user notes*. March 2001.
- [22] A. GEOFFRION, *Generalized benders decomposition*, Journal of Optimization Theory and Applications, **10** (1972), pp. 237–260.
- [23] P. GILL, W. MURRAY, AND M. SAUNDERS, *User's guide for SNOPT 5.3: A Fortran package for large-scale nonlinear programming*, tech. rep., Systems Optimization Laboratory, Stanford University, Stanford, CA, 1997.
- [24] N. GOULD, D. ORBAN, AND P. TOINT, *An interior-point l1-penalty method for nonlinear optimization*, Tech. Rep. RAL-TR-2003-022, Rutherford Appleton Laboratory Chilton, Oxfordshire, UK, November 2003.
- [25] O.K. GUPTA AND A. RAVINDRAN, *Branch and bound experiments in convex nonlinear integer programming*, Management Science, **31**(12) (1985), pp. 1533–1546.
- [26] S. LEYFFER, *Integrating SQP and branch-and-bound for mixed integer nonlinear programming*, Tech. Rep. NA-182, Department of Mathematics, University of Dundee, August 1998.
- [27] Z. LUO, J. STURM, AND S. ZHANG, *Conic convex programming and self-dual embedding*, Optimization Methods and Software, **14** (2000), pp. 169–218.
- [28] G. NEMHAUSER AND L. WOLSEY, *Integer and Combinatorial Optimization*, Wiley, New York, 1988.

- [29] J. NOCEDAL, J. MORALES, R. WALTZ, G. LIU, AND J. GOUX, *Assessing the potential of interior-point methods for nonlinear optimization*, in Large-Scale PDE-Constrained Optimization, Lecture Notes in Computational Science and Engineering, Vol. **30**, 2003, pp. 167–183.
- [30] J. NOCEDAL AND R.A. WALTZ, *Knitro 2.0 user's manual*, Tech. Rep. OTC 02-2002, Optimization Technology Center, Northwestern University, January 2002.
- [31] I. QUESADA AND I. GROSSMANN, *An LP/NLP based branch and bound algorithm for convex MINLP optimization problems*, Computers and Chemical Engineering, **16** (1992), pp. 937–947.
- [32] N. SAHINIDIS, *Baron: A general purpose global optimization software package*, Journal of Global Optimization, **8**(2) (1996), pp. 201–205.
- [33] A. TITS, A. WÄCHTER, S. BAKHTIARI, T. URBAN, AND C. LAWRENCE, *A primal-dual interior-point method for nonlinear programming with strong global and local convergence properties*, SIAM Journal on Optimization, **14** (2003), pp. 173–199.
- [34] R. VANDERBEI, *LOQO user's manual—version 3.10*, Optimization Methods and Software, **12** (1999), pp. 485–514.
- [35] R. VANDERBEI AND D. SHANNO, *An interior-point algorithm for nonconvex nonlinear programming*, Computational Optimization and Applications, **13** (1999), pp. 231–252.
- [36] J. VISWANATHAN AND I. GROSSMAN, *A combined penalty function and outer approximation method for MINLP optimization*, Computers and Chemical Engineering, **14** (1990), pp. 769–782.
- [37] A. WÄCHTER AND L.T. BIEGLER, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Tech. Rep. RC 23149, IBM T.J. Watson Research Center, Yorktown, USA, March 2004.
- [38] T. WESTERLUND AND K. LUNDQVIST, *Alpha-ecp version 5.01: An interactive minlp-solver based on the extended cutting plane method*, Tech. Rep. 01-178-A, Process Design Laboratory at Abo Akademii University, 2001.
- [39] S. ZHANG, *A new self-dual embedding method for convex programming*, Journal of Global Optimization, **29** (2004), pp. 479–496.

PART IV:

EXPRESSION GRAPHS

USING EXPRESSION GRAPHS IN OPTIMIZATION ALGORITHMS

DAVID M. GAY*

Abstract. An expression graph, informally speaking, represents a function in a way that can be manipulated to reveal various kinds of information about the function, such as its value or partial derivatives at specified arguments and bounds thereon in specified regions. (Various representations are possible, and all are equivalent in complexity, in that one can be converted to another in time linear in the expression's size.) For mathematical programming problems, including the mixed-integer nonlinear programming problems that were the subject of the IMA workshop that led to this paper, there are various advantages to representing problems as collections of expression graphs. “Presolve” deductions can simplify the problem, e.g., by reducing the domains of some variables and proving that some inequality constraints are never or always active. To find global solutions, it is helpful sometimes to solve relaxed problems (e.g., allowing some “integer” variables to vary continuously or introducing convex or concave relaxations of some constraints or objectives), and to introduce “cuts” that exclude some relaxed variable values. There are various ways to compute bounds on an expression within a specified region or to compute relaxed expressions from expression graphs. This paper sketches some of them. As new information becomes available in the course of a branch-and-bound (or -cut) algorithm, some expression-graph manipulations and presolve deductions can be revisited and tightened, so keeping expression graphs around during the solution process can be helpful. Algebraic problem representations are a convenient source of expression graphs. One of my reasons for interest in the AMPL modeling language is that it delivers expression graphs to solvers.

Key words. Expression graphs, automatic differentiation, bound computation, constraint propagation, presolve.

AMS(MOS) subject classifications. Primary 68U01, 68N20, 68W30, 05C85.

1. Introduction. For numerically solving a problem, various problem representations are often possible. Many factors can influence one's choice of representation, including familiarity, computational costs, and interfacing needs. Representation possibilities include some broad, often overlapping categories that may be combined with uses of special-purpose libraries: general-purpose programming compiled languages (such as C, C++, Fortran, and sometimes Java), interpreted languages (such as awk, Java, or Python), and “little languages” specialized for dealing with particular problem domains (such as AMPL for mathematical programming or MATLAB for matrix computations — and much else). Common to

*Sandia National Laboratories, Albuquerque, NM 87185-1318. Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. This manuscript (SAND2009-5066C) has been authored by a contractor of the U.S. Government under contract DE-AC04-94AL85000. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

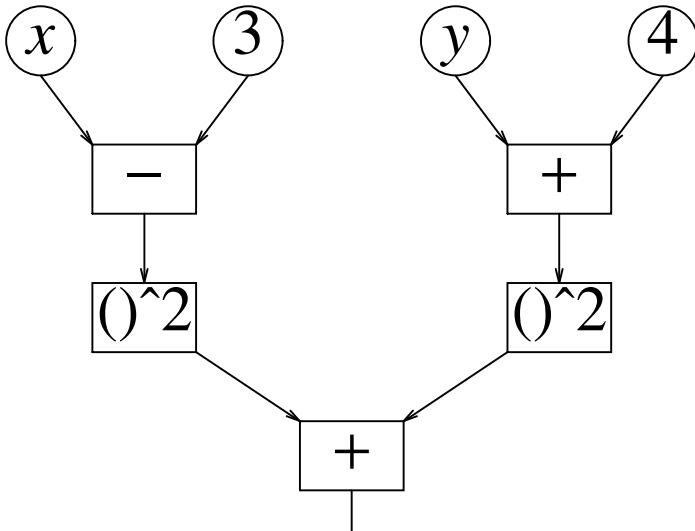


FIG. 1. *Expression graph for $f(x, y) = (x - 3)^2 + (y + 4)^2$.*

most such representations is that they are turned into expression graphs behind the scenes: directed graphs where each node represents an operation, incoming edges represent operands to the operation, and outgoing edges represent uses of the result of the operation. This is illustrated in Figure 1, which shows an expression graph for computing the $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ for computing the function $f(x, y) = (x - 3)^2 + (y + 4)^2$, which involves operators for addition (+), subtraction (-) and squaring ($()^2$).

It can be convenient to have an explicit expression graph and to compute with it or manipulate it in various ways. For example, for smooth optimization problems, we can turn expression graphs for objective and constraint-body evaluations into reasonably efficient ways to compute both these functions and their gradients. When solving mixed-integer nonlinear programming (MINLP) problems, computing bounds and convex underestimates (or concave overestimates) can be useful and can be done with explicit expression graphs. Problem simplifications by “presolve” algorithms and (similarly) domain reductions in constraint programming are readily carried out on expression graphs.

This paper is concerned with computations related to solving a mathematical programming problem: given $D \subseteq \mathbb{R}^n$, $f : D \rightarrow \mathbb{R}$, $c : D \rightarrow \mathbb{R}^m$, and $\ell, u \in D \cup \{-\infty, \infty\}^n$ with $\ell_i \leq u_i \forall i$, find x^* such that $x = x^*$ solves

$$\begin{aligned}
& \text{Minimize } f(x) \\
& \text{subject to } \ell \leq c(x) \leq u \\
& \text{and } x \in D.
\end{aligned} \tag{1.1}$$

For MINLP problems, D restricts some components to be integers, e.g.,

$$D = \mathbb{R}^p \times \mathbb{Z}^q, \tag{1.2}$$

with $n = p + q$.

One of my reasons for interest in the AMPL [7, 8] modeling language for mathematical programming is that AMPL makes explicit expression graphs available to separate solvers. Mostly these graphs are only seen and manipulated by the AMPL/solver interface library [13], but one could also use them directly in the computations described below.

There are various ways to represent expression graphs. For example, AMPL uses a Polish prefix notation (see, e.g., [31]) for the nonlinear parts of problems conveyed to solvers via a “.nl” file. Kearfott [21] uses a representation via 4-tuples (operation, result, left, and right operands). Representations in XML have also been advocated ([9]). For all the specific representations I have seen, converting from one form to another takes time linear in the length (nodes + arcs) of the expression graph.

The rest of this paper is organized as follows. The next several sections discuss derivative computations (§2), bound computations (§3), presolve and constraint propagation (§4), convexity detection (§5), and outer approximations (§6). Concluding remarks appear in the final section (§7).

2. Derivative computations. When f and c in (1.1) are continuously differentiable in their continuous variables (i.e., the first p variables when (1.2) holds), use of their derivatives is important for some algorithms; when integrality is relaxed, partials with respect to nominally integer variables may also be useful (as pointed out by a referee). Similarly, when f and c are twice differentiable, some algorithms (variants of Newton’s method) can make good use of their first and second derivatives. In the early days of computing, the only known way to compute these derivatives without the truncation errors of finite differences was to compute them by the rules of calculus: deriving from, e.g., an expression for $f(x)$ expressions for the components of $\nabla f(x)$, then evaluating the derived formulae as needed. Hand computation of derivatives is an error-prone process, and many people independently discovered [18] a class of techniques called Automatic Differentiation (or Algorithmic Differentiation), called AD below. The idea is to modify a computation so it computes both function and desired partial derivatives as it proceeds — an easy thing to do with an expression graph. Forward AD is easiest to understand and implement: one simply applies the rules of calculus to recur desired partials for the result of an operation from the partials of the operands. When there is

only one independent variable, it is easy and efficient to recur high-order derivatives with respect to that variable. For example, Berz et al. [3, 4] have done highly accurate simulations of particle beams using high-order Taylor series (i.e., by recurring high derivatives).

Suppose f is a function of $n > 1$ variables and that computing $f(x)$ involves L operations. Then the complexity of computing $f(x)$ and its gradient $\nabla f(x)$ by forward AD is $O(nL)$. It can be much faster to use “reverse AD” to compute $f(x)$ and $\nabla f(x)$. With this more complicated AD variant, one first computes $f(x)$, then revisits the operations in reverse order to compute the “adjoint” of each operation, i.e., the partial of f with respect to the result of the operation. By the end of this “reverse sweep”, the computed adjoints of the original variables are the partials of f with respect to these variables, i.e., $\nabla f(x)$. The reverse sweep just involves initializing variables representing the adjoints to zero and adding partials of individual operations times adjoints to the adjoint variables of the corresponding operands, which has the same complexity $O(L)$ as computing $f(x)$. For large n , reverse AD can be much faster than forward AD or finite differences.

The AMPL/solver interface library (ASL) makes arrangements for reverse AD sweeps while reading expression graphs from a “.nl” file and converting them to internal expression graphs. This amounts to a preprocessing step before any numerical computing is done, and is one of many useful kinds of expression-graph walks. Many ways of handling implementation details are possible, but the style I find convenient is to represent each operation (node in the expression graph) by a C “struct” that has a pointer to a function that carries out the operation, pointers to the operands, and auxiliary data that depend on the intended use of the graph. For example, the “expr” structure used in the ASL for binary operations has the fields shown in [Figure 2](#) when only function and gradient computations are allowed [12], and has the more elaborate form shown in [Figure 3](#) when Hessian computations are also allowed [14]. The intent here is not to give a full explanation of these structures, but just to illustrate how representations can vary, depending on their intended uses. In reality, some other type names appear in the ASL, and some fields appear in a different order. [Figures 2](#) and [3](#) both assume typedefs of the form

```
typedef struct expr expr;
typedef double efunc(expr*);
```

so that an “efunc” is a double-valued function that takes one argument, a pointer to an “expr” structure. Use of such a function is illustrated in [Figure 4](#), which shows the ASL’s “op” function for multiplication. This is a particularly simple binary operation in that the left partial is the right operand and vice versa. Moreover the second partials are constants (0 or 1) and need not be computed. In other cases, such as division and

```
struct expr {
    efunc *op;          /* function for this operation */
    int a;              /* adjoint index */
    real dL, dR;        /* left and right partials */
    expr *L, *R;        /* left and right operands */
};
```

FIG. 2. ASL structure for binary operations with only f and ∇f available.

```
struct
expr {
    efunc *op;          /* function for this operation */
    int a;              /* adjoint index (for gradient comp.) */
    expr *fwd, *bak;    /* used in forward and reverse sweeps */
    double d0;          /* deriv of op w.r.t. t in  $x + t*p$  */
    double a0;          /* adjoint (in Hv computation) of op */
    double ad0;         /* adjoint (in Hv computation) of d0 */
    double dL;          /* deriv of op w.r.t. left operand */
    expr *L, *R;        /* left and right operands */
    double dR;          /* deriv of op w.r.t. right operand */
    double dL2;         /* second partial w.r.t. L, L */
    double dLR;         /* second partial w.r.t. L, R */
    double dR2;         /* second partial w.r.t. R, R */
};
```

FIG. 3. ASL structure for binary operations with f , ∇f , and $\nabla^2 f$ available.

the “atan2” function, when Hessian computations are allowed, the function also computes and stores some second partial derivatives.

Once a function evaluation has stored the partials of each operation, the “reverse sweep” for gradient computations by reverse AD takes on a very simple form in the ASL:

$$\begin{aligned} \text{do } *d->a.\text{rp} &+= *d->b.\text{rp} * *d->c.\text{rp}; \\ \text{while}(d = d->\text{next}); \end{aligned} \quad (2.1)$$

Here d points to a “derp” structure (named for *derivative propagation*) of four pointers: $d->a.\text{rp}$ points to an adjoint being updated, $d->b.\text{rp}$ points to an adjoint of the current operation, $d->c.\text{rp}$ points to a partial derivative of this operation, and $d->\text{next}$ points to the next derp structure to be processed. Thus for each of its operands, an operation contributes the product of its adjoint and a partial derivative to the adjoint of the operand.

Hessian or Hessian-vector computations are sometimes useful. Given a vector $v \in \mathbb{R}^n$ and a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ represented by an expression graph of L nodes, we can compute $\nabla^2 f(x)v$ in $O(L)$ operations by what amounts to a mixture of forward and reverse AD, either applying reverse

```

double
f_OPMULT(expr *e A_ASL)
{
    expr *eL = e->L.e;
    expr *eR = e->R.e;
    return  (e->dR = (*eL->op)(eL))
            * (e->dL = (*eR->op)(eR));
}

```

FIG. 4. ASL function for multiplication.

AD to the result of computing $\phi'(0)$ with $\phi(\tau) \equiv f(x + \tau v)$ (computing $\phi(0)$ and $\phi'(0)$ by forward AD), or by applying forward AD to $v^T \nabla f(x)$, with $\nabla f(x)$ computed by reverse AD. Both descriptions lead to the same numerical operations (but have different overheads in Sacado context discussed below). The ASL offers Hessian-vector computations done this way, since some nonlinear optimization solvers use Hessian-vector products in iterative ‘‘matrix-free’’ methods, such as conjugate gradients, for computing search directions.

Many mathematical programming problems (1.1) involve ‘‘partially separable’’ functions. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is partially separable if it has the form

$$f(x) = \sum_i f_i(A_i x),$$

in which the A_i are matrices having only a few rows (varying with i) and n columns, so that f_i is a function of only a few variables. A nice feature of this structure is that f ’s Hessian $\nabla^2 f(x)$ has the form

$$\nabla^2 f(x) = \sum_i A_i^T \nabla^2 f_i(x) A_i,$$

i.e., $\nabla^2 f(x)$ is a sum of outer products involving the little matrices A_i and the Hessians $\nabla^2 f_i(x)$ of the f_i . Knowing this structure, we can compute each $\nabla^2 f_i(x)$ separately with a few Hessian-vector products, then assemble the full $\nabla^2 f(x)$ — e.g., if it is to be used by a solver that wants to see explicit Hessian matrices.

Many mathematical programming problems involve functions having a more elaborate structure called partially-separable structure:

$$f(x) = \sum_i \theta_i \left(\sum_{j=1}^{r_i} f_{ij}(A_{ij} x) \right), \quad (2.2)$$

in which $\theta_i : \mathbb{R} \rightarrow \mathbb{R}$ is smooth and each A_{ij} matrix has only a small number of rows. The full $\nabla^2 f(x)$ is readily assembled from the pieces of

this representation (and their partials). By doing suitable expression-graph walks, the ASL finds partially-separable structure (2.2) automatically and arranges for it to be exploited when explicit Hessians are desired. More graph walks determine the sparsity of the desired Hessians — usually the Hessian of the Lagrangian function. See [14] for more details. (For use in a parallel computing context, I have recently been convinced to add a way to express the Lagrangian function as a sum of pieces and to arrange for efficient computation of the Hessian of each piece, with the sparsity of each piece made available in a preprocessing step.)

The expression-graph walks that the ASL does once to prepare for later numerical evaluations make such computations reasonably efficient, but, as illustrated in the above reverse-sweep loop and in Figure 4, some pointer chasing is still involved. With another kind of graph walk, that done by the *nlc* program described in [13], we can convert expression graphs into Fortran or C (C++), eliminating much of the pointer chasing and some unnecessary operations, e.g., addition of zero and multiplication by ± 1 .

The expression graphs that AMPL uses internally often involve loops, i.e., iterating something over a set, so dealing with loops in expression graphs is not hard. For simplicity in the ASL, the graphs that AMPL writes to “.nl” files to represent nonlinear programming problems are loop-free, with all operations explicit. Perhaps sometime this will change, as it somewhat limits problems that can be encoded in “.nl” files and sometimes makes them much larger than they might be if they were allowed to use looping nodes. This current limitation is somewhat mitigated by an imported-function facility that permits arbitrary functions to be introduced via shared libraries. When such functions are involved in gradient or Hessian computations, the functions must provide first or first and second partials with respect to their arguments, so the ASL can involve the functions in the derivative computations.

Some languages, such as Fortran and C++, allow operator overloading. With overloading, one can use the same arithmetic operators and functions in expressions involving new types; thus, after modifying source code by changing the types of some variables, one can leave the bulk of the source code unchanged and obtain a program with altered (ideally enhanced) behavior. Operator overloading in suitable languages provides another way to make AD computations conveniently available. An early, very general, and often used package for AD computations in C++ codes is ADOL-C [20], which operates by capturing an expression graph (called a “tape” in [20]) as a side effect of doing a computation of interest, then walking the graph to compute the desired derivatives. Because Sandia National Laboratories does much C++ computing and because more specialized implementations are sometimes more efficient, it has been worthwhile to develop our own C++ package, Sacado [2, 33], for AD. The reverse-AD portion of Sacado [15] does a reverse sweep whose core is equivalent to (2.1).

Computations based on operator overloading are very general and convenient, but present a restricted view of a calculation — somewhat like looking through a keyhole. As indicated by the timing results in Table 1 below, when a more complete expression graph is available, it can be used to prepare faster evaluations than are readily available from overloading techniques. Table 1 shows relative and absolute evaluation times for function and gradient evaluations of an empirical energy function for a protein-folding problem considered in [17]. This problem is rich in transcendental function evaluations (such as $\cos()$, $\sqrt{()}$, $\text{atan}()$), which masks some overhead. Computations were on a 3GHz Intel Xeon CPU; the times in the “rel.” column are relative to the time to compute $f(x)$ alone (the “Compiled C, no ∇f ” line), using a C representation obtained with the *nlc* program mentioned above. The “Sacado RAD” line is for C++ code that uses reverse-mode AD via operator overloading provided by Sacado. (Sacado also offers forward-mode AD, which would be considerably slower on this example.) The two ASL lines compare evaluations designed for computing $f(x)$ and $\nabla f(x)$ only (“ASL, fg mode”) with evaluations that save second partials for possible use in computing $\nabla^2 f(x)$ or $\nabla^2 f(x)v$. The “*nlc*” line is for evaluations using C from the *nlc* program; this line excludes time to run *nlc* and compile and link its output. Solving nonlinear mathematical programming problems often involves few enough evaluations that ASL evaluations make the solution process faster than would use of *nlc*, but for an inner loop repeated many times, preparing the inner-loop evaluation with *nlc* (or some similar facility) could be worthwhile.

TABLE 1
Evaluation times for f and ∇f : protein folding ($n = 66$).

Eval style	sec/eval	rel.
Compiled C, no ∇f	2.92e-5	1.0
Sacado RAD	1.90e-4	6.5
<i>nlc</i>	4.78e-5	1.6
ASL, fg mode	9.94e-5	3.4
ASL, pfgh mode	1.26e-4	4.3

One lesson to draw from Table 1 is that while operator overloading is very convenient in large codes, in that one can significantly enhance computations by doing little more than changing a few types, there may be room to improve performance by replacing code involved in computational bottlenecks by alternate code.

Hessian-vector computations provide a more dramatic contrast between evaluations done with operator overloading (in C++) and evaluations prepared with the entire expression graph in view. Table 2 shows timings for Hessian-vector computations done several ways on the Hessian

of a 100×100 dense quadratic form, $f(x) = \frac{1}{2}x^T Qx$. (Such evaluations only involve additions and multiplications and are good for exposing overhead.) The kinds of evaluations in **Table 2** include two ways of nesting the forward (FAD) and reverse (RAD) packages of Sacado, a custom mixture (“RAD2”) of forward- and reverse AD that is also in Sacado, and the “interpreted” evaluations of the AMPL/solver interface library (ASL) prepared by the ASL’s `pfgh_read` routine. The computations were again on a 3GHz Intel Xeon CPU.

TABLE 2
Times for $\nabla^2 f(x)v$ with $f = \frac{1}{2}x^T Qx$, $n = 100$.

Eval style	sec/eval	rel.
RAD \circ FAD	4.70e-4	18.6
FAD \circ RAD	1.07e-3	42.3
RAD2 (Custom mixture)	2.27e-4	9.0
ASL, pfgh mode	2.53e-5	1.0

For much more about AD in general, see Griewank’s book [19] and the “autodiff” web site [1], which has pointers to many papers and packages for AD.

3. Bound computations. Computing bounds on a given expression can be helpful in various contexts. For nonlinear programming in general and mixed-integer nonlinear programming in particular, it is sometimes useful to “branch”, i.e., divide a compact domain into the disjoint union of two or more compact subdomains that are then considered separately. If we find a feasible point in one domain and can compute bounds showing that any feasible points in another subdomain must have a worse objective value, then we can discard that other subdomain.

Various kinds of bound computations can be done by suitable expression graph walks. Perhaps easiest to describe and implement are bound computations based on interval arithmetic [24]: given interval bounds on the operands of an operation, we compute an interval that contains the results of the operation. For example, for any $a \in [\underline{a}, \bar{a}]$ and $b \in [\underline{b}, \bar{b}]$, the product $ab = a \cdot b$ satisfies

$$ab \in [\min(\underline{a}\bar{b}, \underline{a}\bar{b}, \bar{a}\bar{b}), \max(\underline{a}\bar{b}, \underline{a}\bar{b}, \bar{a}\bar{b})].$$

(It is only necessary to compute all four products when $\max(\underline{a}, \bar{b}) < 0$ and $\min(\bar{a}, \bar{b}) > 0$, in which case $ab \in [\min(\underline{a}\bar{b}, \bar{a}\bar{b}), \max(\underline{a}\bar{b}, \bar{a}\bar{b})]$.) When computing with the usual finite-precision floating-point arithmetic, we can use directed roundings to obtain rigorous enclosures.

Unfortunately, when the same variable appears several times in an expression, interval arithmetic treats each appearance as though it could

have any value in its domain, which can lead to very pessimistic bounds. More elaborate interval analysis (see, e.g., [25, 26]) can give much tighter bounds. For instance, *mean-value forms* [25, 28] have an excellent outer approximation property that will be explained shortly. Suppose domain $X \subset \mathbb{R}^n$ is the Cartesian product of compact intervals, henceforth called an *interval vector*, i.e.,

$$X = [\underline{x}_1, \bar{x}_1] \times \cdots \times [\underline{x}_n, \bar{x}_n].$$

Suppose $f : X \rightarrow \mathbb{R}$ and we have a point $c \in X$ and another interval vector $S \subset \mathbb{R}^n$ with the property that

$$\begin{aligned} & \text{for any } x \in X, \text{ there is an } s \in S \\ & \text{such that } f(x) = f(c) + s^T(x - c); \end{aligned} \tag{3.1}$$

if $f \in C^1(\mathbb{R})$, i.e., f is continuously differentiable, it suffices for S to enclose $\{\nabla f(x) : x \in X\}$. Then any enclosure of

$$\{f(c) + s^T(x - c) : x \in X, s \in S\} \tag{3.2}$$

also encloses $f(X) \equiv \{f(x) : x \in X\}$. For an interval vector V with components $[\underline{v}_i, \bar{v}_i]$, define the width $w(V)$ by $w(V) = \max\{\bar{v}_i - \underline{v}_i : 1 \leq i \leq n\}$, and for an interval enclosure $F = [\underline{F}, \bar{F}]$ of $f(X)$, define $\inf\{f(X)\} = \inf\{f(x) : x \in X\}$, $\sup\{f(X)\} = \sup\{f(x) : x \in X\}$, and $\epsilon(F, f(X)) = \max(\inf\{f(X)\} - \underline{F}, \bar{F} - \sup\{f(X)\})$, which is sometimes called the *excess width*. If $f \in C^1(\mathbb{R})$ and $S = \nabla f(X) \equiv \{\nabla f(x) : x \in X\}$, and $F = \{f(c) + s^T(x - c) : s \in S, x \in X\}$, then $\epsilon(F, f(X)) = O(w(X)^2)$, and this remains true if we use interval arithmetic (by walking an expression graph for f) to compute an outer approximation S of $\nabla f(X)$ and compute F from S by interval arithmetic. If $\nabla f(c) \neq 0$, then for small enough $h > 0$ and $X = [c_1 - h, c_1 + h] \times \cdots \times [c_n - h, c_n + h]$, the relative excess width $(w(F) - w(f(X))) / w(X) = O(h)$, which is the excellent approximation property mentioned above. This means that by dividing a given compact domain into sufficiently small subdomains and computing bounds on each separately, we can achieve bounds within a factor of $(1 + \tau)$ of optimal for a specified $\tau > 0$.

We can do better by computing *slopes* [23, 28] rather than interval bounds on ∇f . Slopes are divided differences, and interval bounds on them give an S that satisfies (3.1), so an enclosure of (3.2) gives valid bounds. For $\phi \in C^1(\mathbb{R})$ and $\xi, \zeta \in \mathbb{R}$, the slope $\phi[\xi, \zeta]$ is uniquely defined by

$$\phi[\xi, \zeta] = \begin{cases} (\phi(\xi) - \phi(\zeta)) / (\xi - \zeta) & \text{if } \xi \neq \zeta, \\ \phi'(\zeta) & \text{if } \xi = \zeta. \end{cases}$$

Slopes for functions of n variables are n -tuples of bounds on divided differences; they are not uniquely defined, but can be computed (e.g., from an expression graph) operation by operation in a way much akin to forward

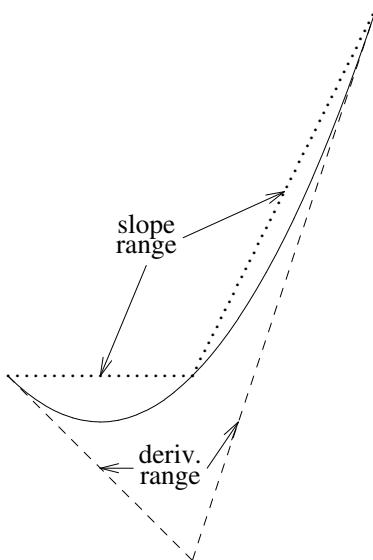


FIG. 5. Slopes and derivatives for $\phi(x) = x^2$, $x \in [-.5, 1.5]$, $c = 0.5$.

AD. The general idea is to compute bounds on $f(X) = \{f(x) : x \in X\}$ by choosing a nominal point z , computing $f(z)$, and using slopes to bound $f(x) - f(z)$ for $x \in X$:

$$f(X) \subseteq \{f(z) + \sum_{i=1}^n s_i(x_i - z_i) : x \in X, s_i \in f[X, z]_i\}$$

where the i -th component $f[X, z]_i$ of an interval slope is a bound on $(f(x + \tau e_i) - f(x))/\tau$ (which is taken to be $\partial f(x)/\partial x_i$ if $\tau = 0$) for $x \in X_i$ and τ such that $x + \tau e_i \in X$. Most simply $X_i = X$ for all i , but we can get tighter bounds [32] at the cost of more memory by choosing

$$X_i = [\underline{x}_i, \bar{x}_i] \times \cdots \times [\underline{x}_i, \bar{x}_i] \times \{c_{i+1}\} \times \cdots \times \{c_n\},$$

e.g., for $c_i \approx \frac{1}{2}(\underline{x}_i + \bar{x}_i)$. With this scheme, permuting the variables may result in different bounds; deciding which of the $n!$ permutations is best might not be easy.

Figure 5 indicates why slopes can give sharper bounds than we get from a first-order Taylor expansion with an interval bound on the derivative. Bounds on $\phi'(X)$ give $S = [-1, 3]$, whereas slope bounds give $S = [0, 2]$.

Sometimes we can obtain still tighter bounds by using second-order slopes [38, 35, 36], i.e., slopes of slopes. The general idea is to compute a slope matrix H such that an enclosure of

$$f(c) + \nabla f(c)(x - c) + (x - c)^T H(x - c)$$

for $x \in X$ gives valid bounds on $f(X)$. (To be rigorous in the face of roundoff error, we must compute interval bounds on $\nabla f(c)$.) Bounds computed this way are sometimes better than those from the mean-value form (3.2). In general, whenever there are several ways to compute bounds, it is usually best to compute all of them and compute their intersection; this is common practice in interval analysis.

As described in [16], I have been modifying some ASL routines to do bound computations from first- and second-order slopes. The computations are similar to forward AD, which greatly simplifies the .nl file reader in that it does not have to set up a reverse sweep. One small novelty is my exploitation of sparsity where possible; the preliminary expression-graph walk done by the .nl reader sets things up so evaluations can proceed more quickly. [Table 3](#) summarizes the bound computations available at this writing, and [Table 4](#) shows the widths of two sets of resulting bounds. In [Table 3](#), F denotes an outer approximation of f . See [16] for details and more references. Explicit expression graphs are convenient for this work, in which the main focus is on properties of particular operations.

TABLE 3
Bound computations.

interval	$F(X) \supset f(X)$
Taylor 1	$f(z) + F'(X)(X - z)$
slope 1	$f(z) + F[X, z](X - z)$
slope 2	$f(z) + f'(z)(X - z)$ + $F[X, z, z](X - z)^2$
slope 2*	slope 2 plus Theorem 2 in [16]

TABLE 4
Bound widths.

Method	Barnes	Sn525
interval	162.417	0.7226
Taylor 1	9.350	0.3609
slope 1	6.453	0.3529
slope 2	3.007	0.1140
slope 2*	2.993	0.1003
true	2.330	0.0903

4. Presolve and constraint propagation. Often it is worthwhile to spend a little time trying to simplify a mathematical programming problem before presenting it to a solver. With the help of suitable bound computations, sometimes it is possible to fix some variables and remove some

constraints. Occasionally a problem can be completely solved this way, but more likely the solver will run faster when presented with a simpler problem.

For linear constraints and objectives, computing bounds is straightforward but best done with directed roundings [6] to avoid false simplifications.

For nonlinear problems, we can use general bounding techniques, such as those sketched in §3, along with specific knowledge about some nonlinear functions, such as that $\sin(x) \in [-1, 1]$ for all x .

Another use of bounding techniques is to reduce variable domains. If we have rigorous bounds showing that part of the nominal problem domain is mapped to values, all of which violate problem constraints, then we can discard that part of the nominal problem domain. In the constraint-propagation community, this is called “constraint propagation”, but it can also be regarded as “nonlinear presolve”. See [34] for more discussion of constraint propagation on expression graphs.

In general, a presolve algorithm may repeatedly revisit bound computations when new information comes along. For instance, if we have upper and lower bounds on all but one of the variables appearing in a linear inequality constraint, we can deduce a bound on the remaining variable; when another deduction implies a tighter bound on one of the other variables, we can revisit the inequality constraint and tighten any bounds deduced from it. Sometimes this leads to a sequence of deductions tantamount to solving linear equations by an iterative method, so it is prudent to limit the repetitions in some way. Similar comments apply when we use nonlinear bounding techniques.

As mentioned in §3, it is sometimes useful to branch, i.e., divide the domain into disjoint subdomains that are then considered separately, perhaps along with the addition of “cuts”, i.e., inequalities that must be satisfied (e.g., due to the requirement that some variables assume integer values). Any such branching and imposition of cuts invites revisiting relevant presolve deductions, which might now be tightened, so an expression-graph representation of the problem can be attractive and convenient.

5. Convexity detection. Convexity is a desirable property for several reasons: it is useful in computing bounds; convex minimization (or concave maximization) problems can be much easier to solve globally than nonconvex ones; and convexity enables use of some algorithms that would otherwise not apply. It is thus useful to know when an optimization problem is convex (or concave).

An approach useful for some problems is to use a problem specification that guarantees convexity; examples include CVXMOD [5] and Young’s recent Ph.D. thesis [37]. More generally, a suitable expression-graph walk [10, 27, 29, 30] can sometimes find sufficient conditions for an expression to be convex or concave. As a special case, some solvers make special provisions for quadratic objectives and sometimes quadratic constraints. Walk-

ing a graph to decide whether it represents a constant, linear, quadratic, or nonlinear expression is straightforward; if quadratic, we can attempt to compute a Cholesky factorization to decide whether it is convex.

6. Outer approximations. Finding outer approximations — convex underestimates and concave overestimates — for a given expression can be useful. By optimizing the outer approximations, we obtain bounds on the expression, and if we compute linear outer approximations (i.e., sets of linear inequalities satisfied by the expression), we can use a linear programming solver to compute bounds, which can be fast or convenient. It is conceptually straightforward to walk an expression graph and derive rigorous outer approximations; see, e.g., [11, 22] for details. Linear outer approximations are readily available from first-order slopes; see §7 of [34].

7. Concluding remarks. Expression graphs are not convenient for users to create explicitly, but are readily derived from high-level representations that are convenient for users. Once we have expression graphs, it is possible to do many useful sorts of computations with them, including creating other representations that are faster to evaluate, carrying out (automatic) derivative computations, computing bounds and outer approximations, detecting convexity, and recognizing problem structure, and simplifying problems. Internal use of expression graphs can be a boon in optimization (and other) algorithms in general, and in mixed-integer nonlinear programming in particular.

Acknowledgment. I thank an anonymous referee for helpful comments.

REFERENCES

- [1] *Autodiff Web Site*, <http://www.autodiff.org>.
- [2] ROSCOE A. BARTLETT, DAVID M. GAY, AND ERIC T. PHIPPS, *Automatic Differentiation of C++ Codes for Large-Scale Scientific Computing*. In Computational Science – ICCS 2006, Vassil N. Alexandrov, Geert Dick van Albada, Peter M.A. Sloot, and Jack Dongarra (eds.), Springer, 2006, pp. 525–532.
- [3] MARTIN BERZ, *Differential Algebraic Description of Beam Dynamics to Very High Orders*, Particle Accelerators **24** (1989), p. 109.
- [4] MARTIN BERZ, YOKO MAKINO, KHODR SHAMSEDDINE, GEORG H. HOFFSTÄTTER, AND WEISHI WAN, *COSY INFINITY and Its Applications in Nonlinear Dynamics*, SIAM, 1996.
- [5] STEPHEN P. BOYD AND JACOB MATTINGLEY, *CVXMOD — Convex Optimization Software in Python*, <http://cvxmod.net/>, accessed July 2009.
- [6] R. FOURER AND D.M. GAY, *Experience with a Primal Presolve Algorithm*. In Large Scale Optimization: State of the Art, W.W. Hager, D.W. Hearn, and P.M. Pardalos (eds.), Kluwer Academic Publishers, 1994, pp. 135–154.
- [7] R. FOURER, D.M. GAY, AND B.W. KERNIGHAN, *A Modeling Language for Mathematical Programming*, Management Science **36**(5) (1990), pp. 519–554.
- [8] ROBERT FOURER, DAVID M. GAY, AND BRIAN W. KERNIGHAN, *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press/Brooks/Cole Publishing Co., 2nd edition, 2003.

- [9] ROBERT FOURER, JUN MA, AND KIPP MARTIN, *An Open Interface for Hooking Solvers to Modeling Systems*, slides for DIMACS Workshop on COIN-OR, 2006, <http://dimacs.rutgers.edu/Workshops/COIN/slides/osil.pdf> .
- [10] R. FOURER, C. MAHESHWARI, A. NEUMAIER, D. ORBAN, AND H. SCHICHL, *Convexity and Concavity Detection in Computational Graphs*, manuscript, 2008, to appear in INFORMS J. Computing.
- [11] EDWARD P. GATZKE, JOHN E. TOLSMA, AND PAUL I. BARTON, *Construction of Convex Function Relaxations Using Automated Code Generation Techniques* Optimization and Engineering **3**, 2002, pp. 305–326.
- [12] DAVID M. GAY, *Automatic Differentiation of Nonlinear AMPL Models*. In Automatic Differentiation of Algorithms: Theory, Implementation, and Application, A. Griewank and G. Corliss (eds.), SIAM, 1991, pp. 61–73.
- [13] DAVID M. GAY, *Hooking Your Solver to AMPL*, AT&T Bell Laboratories, Numerical Analysis Manuscript 93-10, 1993 (revised 1997). <http://www.ampl.com/REFS/hooking2.pdf> .
- [14] DAVID M. GAY, *More AD of Nonlinear AMPL Models: Computing Hessian Information and Exploiting Partial Separability*. In Computational Differentiation : Techniques, Applications, and Tools, Martin Berz, Christian Bischof, George Corliss and Andreas Griewank (eds.), SIAM, 1996, pp. 173–184.
- [15] DAVID M. GAY, *Semiautomatic Differentiation for Efficient Gradient Computations*. In Automatic Differentiation: Applications, Theory, and Implementations, H. Martin Bücker, George F. Corliss, Paul Hovland and Uwe Naumann and Boyana Norris (eds.), Springer, 2005, pp. 147–158.
- [16] DAVID M. GAY, *Bounds from Slopes*, report SAND-1010xxxx, to be available as <http://www.sandia.gov/~dmgay/bounds10.pdf>.
- [17] D.M. GAY, T. HEAD-GORDON, F.H. STILLINGER, AND M.H. WRIGHT, *An Application of Constrained Optimization in Protein Folding: The Poly-L-Alanine Hypothesis*, Frontiers **8**(2) (1992), pp. 4–6.
- [18] ANDREAS GRIEWANK, *On Automatic Differentiation*. In Mathematical Programming: Recent Developments and Applications, M. Iri and K. Tanabe (eds.), Kluwer, 1989, pp. 83–108.
- [19] , ANDREAS GRIEWANK, *Evaluating Derivatives*, SIAM, 2000.
- [20] A. GRIEWANK, D. JUEDES, AND J. UTKE, *Algorithm 755: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math Software **22**(2) (1996), pp. 131–167.
- [21] R. BAKER KEARFOTT *An Overview of the GlobSol Package for Verified Global Optimization*, talk slides, 2002, <http://www.mat.univie.ac.at/~neum/glopt/mss/Kea02.pdf> .
- [22] , PADMANABAN KESAVAN, RUSSELL J. ALLGOR, EDWARD P. GATZKE, AND PAUL I. BARTON, *Outer Approximation Algorithms for Separable Nonconvex Mixed-Integer Nonlinear Programs*, Mathematical Programming **100**(3), 2004, pp. 517–535.
- [23] R. KRAWCZYK AND A. NEUMAIER, *Interval Slopes for Rational Functions and Associated Centered Forms*, SIAM J. Numer. Anal. **22**(3) (1985), pp. 604–616.
- [24] R.E. MOORE, *Interval Arithmetic and Automatic Error Analysis in Digital Computing*, Ph.D. dissertation, Stanford University, 1962.
- [25] RAMON E. MOORE, *Methods and Applications of Interval Analysis*, SIAM, 1979.
- [26] RAMON E. MOORE, R. BAKER KEARFOTT, AND MICHAEL J. CLOUD, *Introduction to Interval Analysis*, SIAM, 2009.
- [27] IVO P. NENOV, DANIEL H. FYLSTRA, AND LUBOMIR V. KOLEV, *Convexity Determination in the Microsoft Excel Solver Using Automatic Differentiation Techniques*, extended abstract, 2004, <http://www.autodiff.org/ad04/abstracts/Nenov.pdf>.
- [28] ARNOLD NEUMAIER, *Interval Methods for Systems of Equations*, Cambridge University Press, 1990.

- [29] DOMINIQUE ORBAN, *Dr. AMPL Web Site*, <http://www.gerad.ca/~orban/drampl/>, accessed July 2009.
- [30] DOMINIQUE ORBAN AND ROBERT FOURER, *Dr. AMPL, A Meta Solver for Optimization*, CORS/INFORMS Joint International Meeting, 2004, <http://users.iems.northwestern.edu/~4er/SLIDES/ban0406h.pdf>.
- [31] *Polish notation*, http://en.wikipedia.org/wiki/Polish_notation, accessed July 2009.
- [32] S.M. RUMP, *Expansion and Estimation of the Range of Nonlinear Functions*, Mathematics of Computation **65**(216) (1996), pp. 1503–1512.
- [33] *Sacado Web Site*, <http://trilinos.sandia.gov/packages/sacado/>.
- [34] HERMANN SCHICHL AND ARNOLD NEUMAIER, *Interval Analysis on Directed Acyclic Graphs for Global Optimization* Journal of Global Optimization **33**(4) (2005), pp. 541–562.
- [35] MARCO SCHNURR, *Steigungen höherer Ordnung zur verifizierten globalen Optimierung*, Ph.D. dissertation, Universität Karlsruhe, 2007.
- [36] MARCO SCHNURR, *The Automatic Computation of Second-Order Slope Tuples for Some Nonsmooth Functions*, Electronic Transactions on Numerical Analysis **30** (2008), pp. 203–223.
- [37] JOSEPH G. YOUNG *Program Analysis and Transformation in Mathematical Programming*, Ph.D. dissertation, Rice University, 2008.
- [38] SHEN ZUHE AND M.A. WOLFE, *On Interval Enclosures Using Slope Arithmetic*, Applied Mathematics and Computation **39** (1990), pp. 89–105.

SYMMETRY IN MATHEMATICAL PROGRAMMING

LEO LIBERTI*

Abstract. Symmetry is mainly exploited in mathematical programming in order to reduce the computation times of enumerative algorithms. The most widespread approach rests on: (a) finding symmetries in the problem instance; (b) reformulating the problem so that it does not allow some of the symmetric optima; (c) solving the modified problem. Sometimes (b) and (c) are performed concurrently: the solution algorithm generates a sequence of subproblems, some of which are recognized to be symmetrically equivalent and either discarded or treated differently. We review symmetry-based analyses and methods for Linear Programming, Integer Linear Programming, Mixed-Integer Linear Programming and Semidefinite Programming. We then discuss a method (introduced in [36]) for automatically detecting symmetries of general (nonconvex) Nonlinear and Mixed-Integer Nonlinear Programming problems and a reformulation based on adjoining symmetry breaking constraints to the original formulation. We finally present a new theoretical and computational study of the formulation symmetries of the Kissing Number Problem.

Key words. MINLP, NLP, reformulation, group, graph isomorphism, permutation, expression tree.

AMS(MOS) subject classifications. 90C11, 90C26, 90C30, 05C25, 20B25.

1. Introduction. Mathematical Programming (MP) is a language for formally describing classes of optimization problems. A MP consists of: parameters, encoding the problem input; decision variables, encoding the problem output; one objective function to be minimized; and a set of constraints describing the feasible set (some of these constraints may be bounds or integrality requirements on the decision variables). The objective function and constraints are described by mathematical expressions whose arguments are the parameters and the decision variables. Let $N = \{1, \dots, n\}$ and $M = \{1, \dots, m\}$ for some nonnegative integers m, n , and $Z \subseteq N$. In general, a MP formulation is as follows:

$$\begin{array}{lll} \min & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & \forall i \in Z \quad x_i \in \mathbb{Z}, \end{array} \quad (1.1)$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are functions that can be written as mathematical expressions involving of a finite number of operators (e.g. $\{+, -, \times, \div, \uparrow, \log, \exp, \sin, \cos, \tan\}$) and x as arguments. If f, g are affine forms and $Z = \emptyset$, (1.1) is a Linear Program (LP). If f, g contain some nonlinear term and $Z = \emptyset$, (1.1) is a Nonlinear Program (NLP), and if f is a convex function and the feasible set $\{x \mid g(x) \leq 0\}$ is convex, (1.1) is a convex NLP (cNLP). If f, g are affine

*LIX, École Polytechnique, F-91128 Palaiseau, France (liberti@lix.polytechnique.fr).

and $Z \neq \emptyset$, (1.1) is a Mixed-Integer Linear Program (MILP), and if $Z = N$ it is an Integer Linear Program (ILP). If f, g contain some nonlinear term and $Z \neq \emptyset$, (1.1) is a Mixed-Integer Nonlinear Program (MINLP), and if f and the feasible set are convex, it is a convex MINLP (cMINLP). A special case of MP, called Semidefinite Programming (SDP) is when f, g are affine forms, $Z = \emptyset$, x is a square matrix, and there is an additional constraint stating that x must be symmetric positive semidefinite ($x \succeq 0$). Although this constraint cannot be written as a mathematical expression of the operators listed above, SDP is important because it can be solved in polynomial time by a special-purpose interior point method [1], and because many tight relaxations of polynomial programming problems can be cast as SDPs.

Symmetries have been used in MP for analysis purposes or in order to speed up solution methods. The general approach is as follows. First, symmetries are detected, either algorithmically or because of some known mathematical property of the given optimization problem. Once a subset of problem symmetries is known, either the MP is reformulated so that some symmetric optima become infeasible and then solved via standard solution methods (static symmetry breaking [43]), or a known solution method is modified so that it recognizes and exploits symmetry dynamically. Symmetries in MP can be broadly classified in two types: *solution* symmetries, i.e. those variable symmetries that fix the set of solutions setwise; and *formulation* symmetries, i.e. those variable symmetries that fix the formulation (encoded in some data structure). If the formulation group structure for a given MP varies considerably from instance to instance, then automatic symmetry detection methods may be required.

Currently, the main effort is that of *removing* symmetries from a problem in order to find a global optimum more quickly. After extensive computational experimentations with all the symmetric instances of most public instance libraries solved by means of Couenne [6] and BARON [57] (both implementing a spatial Branch-and-Bound (sBB) algorithm) and also RECIPE [38] (based on the Variable Neighbourhood Search (VNS) metaheuristic [22]), my own very personal opinion is that removing symmetries is *good* when solving with sBB and *bad* when solving with VNS. The sBB algorithm is a tree search based on bisecting the variable bounds at each tree node along the spatial direction generating the largest error between the solutions of the upper bounding and lower bounding subproblems. The leaf nodes of the search tree contain globally optimal solutions or infeasible portions of space. If the problem has many symmetric optima, a correspondingly large number of leaves contain these optima. When some of the symmetric optima are eliminated from the feasible region, the bound relative to parent nodes is likely to increase, which accelerates sBB convergence. VNS, on the other hand, works by jumping from optimum to optimum via paths defined by local searches started from random points in increasingly large neighbourhoods of the incumbent, attempting to improve

the objective function value. Since in general it is important to explore as much as possible of the feasible region, one usually tends to move to a new optimum even though the objective function value stays the same. In this context, removing symmetries prevents the algorithm from exploring larger portions of the search space.

In this paper, we provide a study of symmetry for NLPs and MINLPs in general form (no convexity assumption is made on f, g). Our literature review (Sect. 2) first presents a general overview of mathematical programming techniques drawing from group theory, particularly in regard to LP, ILP, MILP, SDP, and then focuses on those items that are relevant to automatic symmetry detection (one of the main topics discussed in this paper). An automatic symmetry detection method (originally introduced in [36]) is recounted in Sections 3-4: we construct a digraph that encodes the structure of the mathematical expression representation of f, g , and then apply known graph-based symmetry detection algorithms to derive the group of permutations of the decision variable indices that fix the symbolic structure of f, g . In Sect. 5 we introduce some linear inequalities that are valid for at least one optimum of (1.1) but which are likely to make at least some symmetric optima infeasible. We then present an original application of the proposed techniques to the Kissing Number Problem [30] in Sect. 6: we use our automatic symmetry detection method to formulate a conjecture on the KNP group structure, which we then prove to be true; we derive some symmetry breaking inequalities, and discuss computational results which show the positive impact of the proposed approach.

1.1. Notation. We follow the notation style common in classical algebra, see e.g. [10, 2], with some modifications drawn from computational group theory [60]. Most of the groups considered in this paper act on vectors in \mathbb{R}^n by permuting the components. Permutations act on sets of vectors by acting on each vector in the set. We denote the identity permutation by e . We employ standard group nomenclature: S_n, C_n are the symmetric and cyclic groups of order n . For any function $f : S \rightarrow T$ (where S, T are sets) we denote S (the domain of f) by $\text{dom}(f)$.

For a group $G \leq S_n$ and a set X of row vectors, $XG = \{xg \mid x \in X \wedge g \in G\}$; if Y is a set of column vectors, $GY = \{gy \mid y \in Y \wedge g \in G\}$. If $X = \{x\}$, we denote XG by xG (and similarly GY by Gy if $Y = \{y\}$); xG is also known as the *orbit* of x in G (and similarly for Gy); in computational group theory literature the notation $\text{orb}(x, G)$ is sometimes employed instead of the more algebraic xG . The (setwise) *stabilizer* $\text{stab}(X, G)$ of a set X with respect to a group G is the largest subgroup H of G such that $XH = X$. For any permutation $\pi \in S_n$, let $\Gamma(\pi)$ be the set of its disjoint cycles, so that $\pi = \prod_{\tau \in \Gamma(\pi)} \tau$. For a group G and $\pi \in G$ let $\langle \pi \rangle$ be the subgroup

of G generated by π , and for a subset $S \subseteq G$ let $\langle S \rangle$ be the subgroup of G generated by all elements of S . Given $B \subseteq \{1, \dots, n\}$, $\text{Sym}(B)$ is the symmetric group of all the permutations of elements in B . A permutation

$\pi \in S_n$ is limited to B if it fixes every element outside B ; π acts on $B \subseteq \{1, \dots, n\}$ as a permutation $\rho \in \text{Sym}(B)$ if π fixes B setwise and $\rho = \pi[B]$ is the permutation of B induced by π . Because disjoint cycles commute, it follows from the definition that for all $k \in \mathbb{N}$, $\pi^k[B] = (\pi[B])^k$. A group $G \leq S_n$ with generators $\{g_1, \dots, g_s\}$ acts on $B \subseteq \{1, \dots, n\}$ as H if $\langle g_i[B] \mid i \leq s \rangle = H$; in this case we denote H by $G[B]$. If B is an orbit of the natural action of G on the integers (i.e. the natural action of G on $\bigcup_{\pi \in G} \text{dom}(\pi)$, which fixes every other integer), then it is easy to show that $G[B]$ is a transitive constituent of G [21]. In general, $G[B]$ may not be a subgroup of G : take $G = \langle (1, 2)(3, 4), (1, 3), (4, 2) \rangle$ and $B = \{1, 2\}$, then $G[B] = \langle (1, 2) \rangle \not\leq G$. Let $B, D \subseteq \{1, \dots, n\}$ with $B \cap D = \emptyset$; if $\pi \in S_n$ fixes both B, D setwise, it is easy to show that $\pi[B \cup D] = \pi[B]\pi[D]$.

2. Literature review. This literature review does not only cover the material strictly inherent to later sections, but attempts to be as informative as possible as concerns the use of symmetry in MP, in such a way as to provide an overview which is complementary to [43]. The first part (Sect. 2.1-2.3) of this review covers a representative subset of the most important works about symmetry in optimization, notably in LP, MILP and (briefly) SDP. We survey those topics which are most relevant to later sections (i.e. symmetry detection methods) in Sect. 2.4.

2.1. Symmetry in Linear Programming. The geometrical objects of LP are polyhedra, and there is a very rich literature on symmetric polyhedra [54]. Such results, however, are mostly about the classification of symmetric polyhedra and are rarely used in MP.

The inherent symmetry of the simplex algorithm is studied in [64, 65, 66]. Given two $m \times n$ matrices A, B , let $\mathcal{S}_A = \{x \in \mathbb{R}^{m+n} \mid (I|A)x = 0\}$ (where $(I|A)$ is the $m \times (m+n)$ matrix formed by the $m \times m$ identity followed by the columns of A) and $\mathcal{S}_B = \{x \in \mathbb{R}^{m+n} \mid (I|B)x = 0\}$; A, B are combinatorially equivalent (written $A :: B$) if there exists π in the symmetric group S_{m+n} such that $\pi\mathcal{S}_A = \mathcal{S}_B$. The paper [64] gives different necessary and sufficient conditions for $A :: B$ (among which a formula for constructing all combinatorially equivalent matrices from submatrices of A). In [65] an application to solving matrix games via the simplex method is presented. In [55], Tucker's combinatorial equivalence is used to devise a simplex algorithm variant capable of solving a pair of primal/dual LPs directly without many of the necessary pre-processing steps.

A very recent result [23] shows that every LP has an optimum in the subspace generated by the fixed points of the action of its symmetry group (2.1) on its feasible region.

2.2. Symmetry in Mixed-Integer Linear Programming. The existing work on symmetry in MILP may be classified in three broad categories: (a) the abelian group approach proposed by Gomory to write integer feasibility conditions for Integer Linear Programs (ILPs); (b) symmetry-

breaking techniques for specific problems, whose symmetry group can be computed in advance; (c) general-purpose symmetry group computations and symmetry-breaking techniques to be used in BB-type solution algorithms. We consider MILPs of the form $\min\{cx \mid Ax \leq b \wedge \forall i \in Z \ x_i \in \mathbb{Z}\}$.

Category (a) was established by R. Gomory [20]: given a basis B of the constraint matrix A , it exploits the (abelian) group $\mathcal{G} = \mathbb{Z}^n / \langle \text{col}(B) \rangle$, where \mathbb{Z}^n is the additive group of integer n -sequences and $\langle \text{col}(B) \rangle$ is the additive group generated by the columns of the (nonsingular) matrix B . Consider the natural group homomorphism $\varphi : \mathbb{Z}^n \rightarrow \mathcal{G}$ with $\ker \varphi = \langle \text{col}(B) \rangle$: letting (x_B, x_N) be a basic/nonbasic partition of the decision variables, apply φ to the standard form constraints $Bx_B + Nx_N = b$ to obtain $\varphi(Bx_B) + \varphi(Nx_N) = \varphi(b)$. Since $\varphi(Bx_B) = 0$ if and only if $x_B \in \mathbb{Z}^n$, setting $\varphi(Nx_N) = \varphi(b)$ is a necessary and sufficient condition for x_B to be integer feasible. Gomory's seminal paper gave rise to further research, among which [69, 5]. The book [25] is a good starting point.

Category (b) is possibly the richest in terms of number of published papers. Many types of combinatorial problems exhibit a certain amount of symmetry. Symmetries are usually broken by means of specific branching techniques (e.g. [41]), appropriate global cuts (e.g. [61]) or special formulations [31, 9] based on the problem structure. The main limitation of the methods in this category is that they are difficult to generalize and/or to be made automatic.

Category (c) contains two main research streams. The first was established by Margot in the early 2000s [39, 40], and is applicable to Binary Linear Programs (BLPs) in the form:

$$\begin{array}{lll} \min & cx \\ Ax & \leq b \\ x & \in \{0, 1\}^n. \end{array} \quad \left. \right\}$$

Margot [39, 43] defines the *relaxation group* $G^{\text{LP}}(P)$ of a BLP P as:

$$G^{\text{LP}}(P) = \{\pi \in S_n \mid c\pi = c \wedge \exists \sigma \in S_n \ (\sigma b = b \wedge \sigma A\pi = A)\}, \quad (2.1)$$

or, in other words, all relabellings of problem variables for which the objective function and constraints are the same. The relaxation group (2.1) is used to derive effective BB pruning strategies by means of isomorphism pruning and isomorphism cuts local to some selected BB tree nodes (Margot extended his work to general integer variables in [42]). Further results along the same lines (named *orbital branching*) are obtained for covering and packing problems in [50, 51]: if O is an orbit of some subgroup of the relaxation group, at each BB node the disjunction $(\bigvee_{i \in O} x_i = 1) \vee \sum_{i \in O} x_i = 0$ induces a feasible division of the search space; orbital branching restricts this disjunction to $x_h = 1 \vee \sum_{i \in O} x_i$ where h is an arbitrary index in O .

The second was established by Kaibel et al. in 2007 [26, 15], with the introduction of the packing and partitioning orbitopes, i.e. convex hulls

of certain 0-1 matrices that represent possible solutions to sets of packing and partitioning constraints. These are used in problems defined in terms of matrices of binary decision variables x_{ij} (for $i \leq m, j \leq n$). Since a typical packing constraint is $\sum_{j \in J_i} x_{ij} \leq 1$ for some $i \leq m, J_i \subseteq \{1, \dots, n\}$ (partitioning constraints simply replace inequalities with equations), sets of such constraint may exhibit column symmetries in $\prod_{i \leq m} \text{Sym}(J_i)$, and row symmetries in S_m . Orbitopes are convex hulls of binary $m \times n$ matrices that have lexicographically ordered columns: their vertices represent a subset of feasible solutions of the corresponding packing/partitioning problem from which several symmetries have been removed. Given a partition C_1, \dots, C_q of the variable indices, a permutation $\pi \in G^{\text{LP}}(P)$ is an *orbitopal symmetry* if there are $p, r \leq q$ such that π is a bijection $C_p \rightarrow C_r$ that keeps all other C_s elementwise fixed, for $s \notin \{p, r\}$ [7]. In [26], a complete description of packing/partitioning orbitopes in terms of linear inequalities is provided ([15] gives a much shorter, more enlightening and less technical presentation than that given in [26]). Inspired by the work on orbitopes, E. Friedman proposed a similar but more general approach leading to *fundamental domains* [17]: given a feasible polytope $X \subseteq [0, 1]^n$ with integral extreme points and a group G acting as an affine transformation on X (i.e. for all $\pi \in G$ there is a matrix $A \in GL(n)$ and an n -vector d such that $\pi x = Ax + d$ for all $x \in X$), a fundamental domain is a subset $F \subset X$ such that $GF = X$.

2.3. Symmetry in Semidefinite Programming. There are several works describing the exploitation of symmetry in Semidefinite Programming (see e.g. [27, 19, 28]). Much of the material in this section is taken from the commendable tutorial [68]. Consider the following SDP:

$$\begin{array}{lll} \min_X & C \bullet X \\ \forall k \leq m & A_k \bullet X \leq b_i \\ & X \succeq 0, \end{array} \quad \left. \right\} \quad (2.2)$$

where X is an $n \times n$ symmetric matrix abd $M_1 \bullet M_2 = \text{trace}(M_1^\top M_2)$ is the trace product between matrices M_1, M_2 . Let G^{SDP} be the largest subgroup of S_n such that if X^* is an optimum then πX^* is also an optimum for all $\pi \in G^{\text{SDP}}$, where the action of π on an $n \times n$ matrix M is to permute the columns *and* the rows of M according to π . If X^* is an optimum, taking $\frac{1}{|G^{\text{SDP}}|} \sum_{\pi \in G} \pi X^*$ shows that there is always an optimal solution of (2.2) in

\mathcal{B} , the space of G^{SDP} -invariant matrices. Let R_1, \dots, R_k be the orbits of $\{(i, j) \mid i, j \leq n\}$ under G^{SDP} , and for all $r \leq k$ let $B^r = (b_{ij}^r)$ the 0-1 incidence matrix of $(i, j) \in R_r$ (i.e. $b_{ij}^r = 1$ of $(i, j) \in R_r$ and 0 otherwise). Then B^1, \dots, B^k is a basis of \mathcal{B} and (2.2) can be re-cast as a search over the coefficients of a linear form in B^1, \dots, B^k :

$$\left. \begin{array}{l} \min_y \quad \sum_{j \leq k} (C \bullet B^j) y_j \\ \forall i \leq m \quad \sum_{j \leq k} (A_i \bullet B^j) y_j = b_i \\ \sum_{j \leq k} y_j B^j \succeq 0. \end{array} \right\} \quad (2.3)$$

By rewriting (2.2) and (2.3) over \mathbb{C} , \mathcal{B} becomes a semisimple algebra over \mathbb{C} . This implies that it is possible to find an algebra isomorphism

$$\phi : \mathcal{B} \rightarrow \bigoplus_{t \leq d} \mathbb{C}^{m_t \times m_t}$$

for some integers d and m_t ($t \leq d$). This allows a size reduction of the SDP being solved, as the search only needs to be conducted on the smaller-dimensional space spanned by $\phi(\mathcal{B})$.

A different line of research is pursued in [27]: motivated by an application (truss optimization), it is shown that the barrier subproblem of a typical interior point method for SDP “inherits” the same symmetries as the original SDP.

2.4. Automatic symmetry detection. Automatic symmetry detection does not appear prominently in the mathematical programming literature. A method for finding the MILP relaxation group (2.1), based on solving an auxiliary MILP encoding the condition $\sigma A \pi = A$, was proposed and tested in [33] (to the best of our knowledge, the only approach for symmetry detection that does not reduce the problem to a graph). A more practically efficient method consists in finding the automorphism group of vertex-colored bipartite graph encoding the incidence of variables in constraints. If the symmetry π is orbitopal and the system $Ax \leq b$ contains at least a *leading constraint*, i.e. a π -invariant constraint that has exactly one nonzero column in each C_p (for $p \leq q$) then a set of generators for $G^{\text{LP}}(P)$ can be found in linear time in the number of nonzeros of A [7].

The Constraint Programming (CP) literature contains many papers on symmetries. Whereas most of them discuss symmetry breaking techniques, a few of them deal with automatic symmetry detection and are relevant to the material presented in the rest of the paper; all of them rely on reducing the problem to a graph and solving the associated GRAPH ISOMORPHISM (GI) problem. In CP, symmetries are called *local* if they hold at a specific search tree node, and *global* otherwise. Solution symmetries are also called *semantic* symmetries, and formulation symmetries are also called *syntactic* or *constraint* symmetries. A Constraint Satisfaction Problem (CSP) can be represented by its *microstructure complement*, i.e. a graph whose vertices are assignments $x = a$ (where x ranges over all CSP variables and a over all values in the domain of x), and whose edges $(x_i = a, x_j = b)$ indicate that the two assignments $x_i = a$ and $x_j = b$ are incompatible either because of a constraint in the CSP or because $i = j$ and $a \neq b$. Constraint symmetries

are defined in [11] as the automorphisms of the microstructure complement. A *k-ary nogood* is a *k*-partial solution (i.e. an assignment of values to *k* variables) which cannot be extended to a full solution of the given CSP instance. The *k-nogood hypergraph* of the CSP has assignments $x = a$ as vertices and all *m*-ary nogoods as hyperedges, for $m \leq k$. For a *k*-ary CSP (one whose constraints have maximum arity *k*), the group of solution symmetries is equal to the automorphism group of its *k*-nogood hypergraph [11]. In [13] (possibly the first work in which a reduction from formulation-type symmetries to GI was proposed), SAT symmetries are automatically detected by reducing the problem to a bipartite graph, and identified by solving the corresponding GI instance, similarly to the approach taken in [50]. In [53], constraints involving the arithmetic operations $+, -, \times$ are reduced to Directed Acyclic Graphs (DAG) whose leaf vertices represent variables and intermediate vertices represent operators; vertex colors identify same operator types and constraints having the same right hand sides. Thm. 3.1 in [53] shows that the automorphism group of this DAG is isomorphic to the constraint group of the corresponding CSP instance, and might be considered the CP equivalent of Thm. 4.1 and Thm. 4.2 appearing below (although the proof techniques are completely different). In [52], a systematic reduction of many types of constraints to an equivalent graph form is proposed; an improved representation and extensive computational results are given in [46]. The problem of determining the constraint group of a model (instead of an instance) is discussed in [47] — we pursue a similar line of reasoning when inferring the structure of the KNP group (independently of the instance) from a sequence of automatically computed KNP instance groups.

3. Groups of a mathematical program. Let P be a MP with formulation as in (1.1), and $\mathcal{F}(P)$ (resp. $\mathcal{G}(P)$) be the set of its feasible (resp. globally optimal) points. Two important groups are connected with P . The *solution group* is the group of all permutations of the variable indices which map $\mathcal{G}(P)$ into itself; it is defined formally as $G^*(P) = \text{stab}(\mathcal{G}(P), S_n)$ and contains as a subgroup the “symmetry group” of P , defined limited to MILPs in [43] as the group of permutations mapping feasible solutions into feasible solutions having the same objective function value. Computing solution groups directly from their definition would imply knowing $\mathcal{G}(P)$ aprioristically, which is evidently unrealistic.

The other important group related to P (denoted by \bar{G}_P) fixes the formulation of P . For two functions $f_1, f_2 : \mathbb{R}^n \rightarrow \mathbb{R}$ we write $f_1 = f_2$ to mean $\text{dom}(f_1) = \text{dom}(f_2) \wedge \forall x \in \text{dom}(f_1) (f_1(x) = f_2(x))$. Then

$$\bar{G}_P = \{\pi \in S_n \mid Z\pi = Z \wedge f(x\pi) = f(x) \wedge \exists \sigma \in S_m (\sigma g(x\pi) = g(x))\}.$$

It is easy to show that $\bar{G}_P \leq G^*(P)$: let $\pi \in \bar{G}_P$ and $x^* \in \mathcal{G}(P)$; $x^*\pi \in \mathcal{F}(P)$ because $Z\pi = Z$, $g(x^*\pi) = \sigma^{-1}g(x^*)$; and it has the same function

value because $f(x^*\pi) = f(x^*)$ by definition. Thus $\mathcal{G}(P)\pi = \mathcal{G}(P)$ and $\pi \in G^*(P)$.

The definition of \bar{G}_P implies the existence of a method for testing whether $f(x\pi) = f(x)$ and whether there is a permutation $\sigma \in S_m$ such that $\sigma g(x\pi) = g(x)$. Since NONLINEAR EQUATIONS (determining if a set of general nonlinear equations has a solution) is an undecidable problem in general [70], such tests are algorithmically intractable. Instead, we assume the existence of a YES/NO oracle \equiv that answers YES if it can establish that $f_1 = f_2$ (i.e. f_1, f_2 have the same domain and are pointwise equal on their domain). Such an oracle defines an equivalence relation \equiv on the set of all functions appearing in (1.1): if a pair of functions (f_1, f_2) belongs to the relation then the functions are equal, but not all pairs of equal functions might belong to \equiv (i.e. \equiv might answer NO even though $f_1 = f_2$). This weakening of the equality relationship will allow us to give an algorithmically feasible definition of the symmetry group of the formulation.

We define the \equiv oracle by only considering functions that can be written syntactically using infix notation in terms of a finite set of operators (e.g. arithmetic, logarithm, exponential and so on), a finite set of constants in \mathbb{Q} and the set of problem variables x_1, \dots, x_n . Such functions can be naturally represented by means of expression trees (Fig. 1 left) which, by contracting leaf vertices with equal labels, can be transformed into DAGs as shown in Fig. 1 (right). The \equiv oracle is then implemented as a recur-

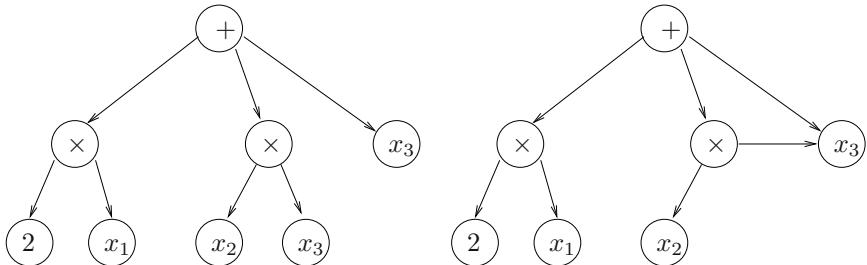


FIG. 1. Expression tree for $2x_1 + x_2x_3 + x_3$ (left). Equal variable vertices can be contracted to obtain a DAG (right).

sive graph exploration. The function DAG representation is well known and should perhaps be attributed to folklore (it is mentioned e.g. in [29], Sect. 2.3). DAG representable functions are routinely used in Global Optimization (GO) to automatically build relaxations of MINLPs [62, 32, 6] or tighten the variable bounds [58]. In the context of symmetry in MP, precise definitions for DAG representable functions and the \equiv oracle implementation are given in [36, 12]. The *formulation group* of P can now be defined as:

$$G_P = \{\pi \in S_n | Z\pi = Z \wedge f(x\pi) \equiv f(x) \wedge \exists \sigma \in S_m (\sigma g(x\pi) \equiv g(x))\}. \quad (3.1)$$

Because for any function h , $h(x\pi) \equiv h(x)$ implies $h(x\pi) = h(x)$ for all $x \in \text{dom}(h)$, it is clear that $G_P \leq \bar{G}_P$. Thus, it also follows that $G_P \leq G^*(P)$. Although \bar{G}_P is defined for any MINLP (1.1), if P is a BLP, then $\bar{G}_P = G^{\text{LP}}(P)$ [36]. We remark that if f_1, f_2 are linear forms, then $f_1 = f_2$ implies $f_1 \equiv f_2$. In other words, for linear forms, \equiv and $=$ are the same relation [36]. As a corollary, if P is a BLP, then $G_P = G^{\text{LP}}(P)$.

If a set of mathematical functions share the same arguments, as for the objective function f and constraints g of (1.1), the corresponding DAGs for f, g_1, \dots, g_m can share the same variable leaf vertices. This yields a DAG $D_P = (V_P, A_P)$ (formed by the union of all the DAGs of functions in P followed by the contraction of leaf vertices with same variable index label) which represents the mathematical structure P [49, 58].

4. Automatic computation of the formulation group. The method proposed in this section also appears (with more details) in [36]. As mentioned in the literature review, similar techniques are available in CP [53].

We first define an equivalence relation on V_P which determines the interchangeability of two vertices of D_P . Let \mathcal{S}_F be the singleton set containing the root vertex of the objective function, \mathcal{S}_C of all constraint root vertices, \mathcal{S}_O of all vertices representing operators, \mathcal{S}_K of all constant vertices and \mathcal{S}_V of all variable vertices. For $v \in \mathcal{S}_F$, we denote optimization direction of the corresponding objective function by $d(v)$; for $v \in \mathcal{S}_C$, we denote the constraint sense by $s(v)$. For $v \in \mathcal{S}_O$, we let $\ell(v)$ be the level of v in D_P , i.e. the length of the path from the root to v (ℓ is well defined as the only vertices with more than one incoming arc are the leaf vertices), $\lambda(v)$ be its operator label and $o(v)$ be the order of v as an argument of its parent vertex if the latter represents a noncommutative operator, or 1 otherwise. For $v \in \mathcal{S}_K$, we let $\mu(v)$ be the value of v . For $v \in \mathcal{S}_V$ we let $r(v)$ be the 2-vector of lower and upper variable bounds for v and $\zeta(v)$ be 1 if v represents an integral variable or 0 otherwise. We now define the relation \sim on V_P as follows

$$\begin{aligned} \forall u, v \in V_P \quad u \sim v \Leftrightarrow & (u, v \in \mathcal{S}_F \wedge d(u) = d(v)) \\ & \vee (u, v \in \mathcal{S}_C \wedge s(u) = s(v)) \\ & \vee (u, v \in \mathcal{S}_O \wedge \ell(u) = \ell(v) \wedge \lambda(u) = \lambda(v) \wedge o(u) = o(v)) \\ & \vee (u, v \in \mathcal{S}_K \wedge \mu(u) = \mu(v)) \\ & \vee (u, v \in \mathcal{S}_V \wedge r(u) = r(v) \wedge \zeta(u) = \zeta(v)). \end{aligned}$$

It is easy to show that \sim is an equivalence relation on V_P , and therefore partitions V_P into K disjoint subsets V_1, \dots, V_K .

For a digraph $D = (V, A)$, its automorphism group $\text{Aut}(D)$ is the group of vertex permutations γ such that $(\gamma(u), \gamma(v)) \in A$ for all $(u, v) \in A$ [56]. Let $G^{\text{DAG}}(P)$ be the largest subgroup of $\text{Aut}(D_P)$ fixing V_k setwise for all $k \leq K$. We assume without loss of generality that the vertices of D_P are

uniquely numbered so that for all $j \leq n$, the j -th vertex corresponds to the leaf vertex for variable x_j (the rest of the numbering is not important), i.e. $\mathcal{S}_V = \{1, \dots, n\}$.

Let $G \leq S_n$ and ω be a subset of $\{1, \dots, n\}$. Let $H = \text{Sym}(\omega)$ and define the mapping $\psi : G \rightarrow H$ by $\psi(\pi) = \pi[\omega]$ for all $\pi \in G$. Then the following holds.

THEOREM 4.1 ([36], Thm. 4). *ψ is a group homomorphism if and only if G stabilizes ω setwise.*

Next, we note that $G^{\text{DAG}}(P)$ fixes \mathcal{S}_V setwise [36]. As a corollary to Thm. 4.1, the map $\varphi : G^{\text{DAG}}(P) \rightarrow \text{Sym}(\mathcal{S}_V)$ given by $\varphi(\gamma) = \gamma[\mathcal{S}_V]$ is a group homomorphism.

THEOREM 4.2 ([36], Thm. 7). *$\text{Im } \varphi = G_P$ groupwise.*

By Thm. 4.2, we can automatically generate G_P by looking for the largest subgroup of $\text{Aut}(D_P)$ fixing all V_k 's. Thus, the problem of computing G_P has been reduced to computing the (generators of the) automorphism group of a certain vertex-coloured DAG. This is in turn equivalent to the GI problem [3]. GI is in **NP**, but it is not known whether it is in **P** or **NP**-complete. A notion of GI-completeness has therefore been introduced for those graph classes for which solving the GI problem is as hard as solving it on general graphs [67]. Rooted DAGs are GI-complete [8] but there is an algorithm for solving the GI problem on trees which is linear in the number of vertices in the tree ([56], Ch. 8.5.2). This should give an insight as to the type of difficulty inherent to computing $\text{Aut}(D_P)$.

COROLLARY 4.1. *If C' is a set of group generators of $G^{\text{DAG}}(P)$, then $C = \{\pi[\mathcal{S}_V] \mid \pi \in C'\}$ is a set of generators for G_P .*

Cor. 4.1 allows the practical computation of a formulation group: one first forms the graph D_P , then computes generators C' of $G^{\text{DAG}}(P)$, and finally considers their action on \mathcal{S}_V to explicitly construct C . Based on the results of this section, we implemented a software system (called **symgroup**) that automatically detects the formulation group of a problem (1.1). Our system first calls AMPL [16] to parse the instance; the ROSE Reformulation/Optimization Software Engine [37] AMPL-hooked solver is then called (with ROSE's **Rsymgroup** reformulator) to produce a file representation of the problem expression DAG. This is then fed into *nauty*'s [45, 44] **dreadnaut** shell to efficiently compute the generators of $\text{Aut}(D_P)$. A system of shell scripts and Unix tools parses the *nauty* output to form a valid GAP [18] input, used to print the actual group description via the command **StructureDescription**.

5. Symmetry breaking constraints. Once the formulation group is detected, we can adjoin constraints to (1.1) in order to make some of the symmetric optima infeasible. According to the classification in [35], this is a reformulation of the narrowing type.

DEFINITION 5.1. *Given a problem P , a narrowing Q of P is a formulation (1.1) such that (a) there is a function $\eta : \mathcal{F}(Q) \rightarrow \mathcal{F}(P)$ for which*

$\eta(\mathcal{G}(Q))$ (the image of $\mathcal{G}(Q)$ under η) is a subset of $\mathcal{G}(P)$, and (b) Q is infeasible only if P is.

Our narrowing rests on adjoining some static symmetry breaking inequalities (SSBIs) [43] to the original formulation, i.e. inequalities that are designed to cut some of the symmetric solutions while keeping at least one optimal one. The reformulated problem can then be solved by standard software packages such as CPLEX [24] (for MILPs) and COUENNE [6] or BARON [57] for MINLPs.

We first give a formal definition of SSBIs that makes them depend on a group rather than just a set of solutions.

DEFINITION 5.2. *Given a permutation $\pi \in S_n$ acting on the component indices of the vectors in a given set $X \subseteq \mathbb{R}^n$, the constraints $g(x) \leq 0$ (that is, $\{g_1(x) \leq 0, \dots, g_q(x) \leq 0\}$) are symmetry breaking constraints (SBCs) with respect to π and X if there is $y \in X$ such that $g(y\pi) \leq 0$. Given a group G , $g(x) \leq 0$ are SBCs w.r.t G and X is there is $y \in XG$ such that $g(y) \leq 0$.*

If there are no ambiguities as regards X , we simply say “SBCs with respect to π ” (respectively, G). In most cases, $X = \mathcal{G}(P)$. The following facts are easy to prove.

1. For any $\pi \in S_n$, if $g(x) \leq 0$ are SBCs with respect to π, X then they are also SBCs with respect to $\langle \pi \rangle, X$.
2. For any $H \leq G$, if $g(x) \leq 0$ are SBCs with respect to H, X then they are also SBCs with respect to G, X .
3. Let $g(x) \leq 0$ be SBCs with respect to $\pi \in S_n, X \subseteq \mathbb{R}^n$ and let $B \subseteq \{1, \dots, n\}$. If $g(x) \equiv g(x[B])$ (i.e. the constraints g only involve variable indices in B) then $g(x) \leq 0$ are also SBCs with respect to $\pi[B], X[B]$.

As regards Fact 3, if $g(x) \equiv g(x[B])$ we denote the SBCs $g(x) \leq 0$ by $g[B](x) \leq 0$; if B is the domain of a permutation $\alpha \in \text{Sym}(B)$, we also use the notation $g[\alpha](x) \leq 0$.

EXAMPLE 1. Let $y = (1, 1, -1)$, $X = \{y\}$ and $\pi = (1, 2, 3)$; then $\{x_1 \leq x_2, x_1 \leq x_3\}$ are SBCs with respect to π and X because $y\pi$ satisfies the constraints. $\{x_1 \leq x_2, x_2 \leq x_3\}$ are SBCs with respect to S_3 and X because $(-1, 1, 1) = y(1, 2, 3) \in X S_3$; however, they are not SBCs with respect to $\langle (2, 3) \rangle$ and X because $X \langle (2, 3) \rangle = \{y, y(2, 3)\} = \{(1, 1, -1), (1, -1, 1)\}$ and neither vector satisfies the constraints.

We use SBCs to yield narrowings of the original problem P .

THEOREM 5.1 ([36], Thm. 11). *If $g(x) \leq 0$ are SBCs for any subgroup G of G_P and $\mathcal{G}(P)$, then the problem Q obtained by adjoining $g(x) \leq 0$ to the constraints of P is a narrowing of P .*

6. An application to the Kissing Number Problem. Given positive integers D, N , the decision version of the KISSING NUMBER PROBLEM (KNP) [30] asks whether N unit spheres can be positioned adjacent to a unit sphere centered in the origin in \mathbb{R}^D . The optimization version asks

for the maximum possible N . The pedigree of this problem is illustrious, having originated in a discussion between I. Newton and D. Gregory. The name of the problem is linked to billiard game jargon: when two balls touch each other, they are said to “kiss”. As both Newton and Gregory were of British stock, one may almost picture the two chaps going down the pub arm in arm for a game of pool and a pint of ale; and then, in the fumes of alcohol, getting into a brawl about whether twelve or thirteen spheres might kiss a central one if the billiard table surface was tridimensional. This interpretation disregards the alleged scholarly note (mentioned in [63]) about the problem arising from an astronomical question. When $D = 2$, the maximum feasible N is of course 6 (hexagonal lattice). When $D = 3$, the maximum feasible N was conjectured by Newton to be 12 and by Gregory to be 13 (Newton was proven right 180 years later [59]). The problem for $D = 4$ was settled recently with $N = 24$ [48]. The problem for $D = 5$ is still open: a lower bound taken from lattice theory is 40, and an upper bound derived with Bachoc and Vallentin’s extension [4] of Delsarte’s Linear Programming (LP) bound [14] is 45.

We formulate the decision version of the KNP as a nonconvex NLP:

$$\left. \begin{array}{ll} \max_{x, \alpha} & \alpha \\ \forall i \leq N & \|x^i\|^2 = 4 \\ \forall i < j \leq N & \|x^i - x^j\|^2 \geq 4\alpha \\ \forall i \leq N & x^i \in [-2, 2]^D \\ & \alpha \in [0, 1]. \end{array} \right\} \quad (6.1)$$

For any given $N, D > 1$, if a global optimum (x^*, α^*) of (6.1) has $\alpha^* = 1$ then a kissing configuration of N balls in \mathbb{R}^D exists; otherwise, it does not. In practice, (6.1) is usually solved by heuristics such as Variable Neighbourhood Search (VNS) [30], because solving it by sBB takes too long even on very small instances. One of the reasons for the slow convergence of sBB is that (6.1) has many symmetries. In fact, $\text{Aut}(\mathcal{G}(\text{KNP}))$ has infinite (uncountable) cardinality: each optimum x^* can be rotated by any angle in \mathbb{R}^D , and hence for all orthogonal transformations $\mu \in SO(D, \mathbb{R})$ (the special orthogonal group of \mathbb{R}^D), $\mu(x^*) \in \mathcal{G}(\text{KNP})$. Such symmetries can be easily disposed of by deciding the placement of D spheres so that they are mutually adjacent as well as adjacent to the central sphere in \mathbb{R}^D , but computational experience suggests that this does little, by itself, to decrease the size of the sBB tree.

We used the `symmgroup` system in order to detect the structure of $G_{(6.1)}$ automatically for a few KNP instances, obtaining an indication that $G_{(6.1)} \cong S_D$. However, since D is small with respect to N , this is not likely to help the solution process significantly. Let $x^i = (x_{i1}, \dots, x_{iD})$ for all $i \leq N$. As in [30] we remark that, for all $i < j \leq N$:

$$\|x^i - x^j\|^2 = \sum_{k \leq D} (x_{ik} - x_{jk})^2 = 8 - 2 \sum_{k \leq D} x_{ik} x_{jk}, \quad (6.2)$$

because $\sum_{k \leq D} x_{ik}^2 = \|x^i\|^2 = 4$ for all $i \leq N$. Let Q be (6.1) reformulated according to (6.2): automatic detection of G_Q yields an indication that $G_Q \cong S_D \times S_N$, which is a considerably larger group. The difference lies in the fact that the binary minus is in general not commutative; however, it *is* commutative whenever it appears in terms like $\|x^i - x^j\|$ (by definition of Euclidean norm). Since automatic symmetry detection is based on expression trees, commutativity of an operator is decided at the vertex representing the operator, rather than at the parent vertex. Thus, on (6.1), our automatic system fails to detect the larger group. Reformulation (6.2) prevents this from happening, thereby allowing the automatic detection of the larger group.

EXAMPLE 2. Consider the KNP instance defined by $N = 6, D = 2$, whose variable mapping

$$\begin{pmatrix} x_{11} & x_{12} & x_{21} & x_{22} & x_{31} & x_{32} & x_{41} & x_{42} & x_{51} & x_{52} & x_{61} & x_{62} & \alpha \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 & y_{10} & y_{11} & y_{12} & y_{13} \end{pmatrix}$$

yields the following flat [32] instance:

$$\begin{array}{lll} \min(-y_{13}) & 2y_{13} + y_1y_3 + y_2y_4 \leq 4 & 2y_{13} + y_3y_{11} + y_4y_{12} \leq 4 \\ y_1^2 + y_2^2 = 4 & 2y_{13} + y_1y_5 + y_2y_6 \leq 4 & 2y_{13} + y_5y_7 + y_6y_8 \leq 4 \\ y_3^2 + y_4^2 = 4 & 2y_{13} + y_1y_7 + y_2y_8 \leq 4 & 2y_{13} + y_5y_9 + y_6y_{10} \leq 4 \\ y_5^2 + y_6^2 = 4 & 2y_{13} + y_1y_9 + y_2y_{10} \leq 4 & 2y_{13} + y_5y_{11} + y_6y_{12} \leq 4 \\ y_7^2 + y_8^2 = 4 & 2y_{13} + y_1y_{11} + y_2y_{12} \leq 4 & 2y_{13} + y_7y_9 + y_8y_{10} \leq 4 \\ y_9^2 + y_{10}^2 = 4 & 2y_{13} + y_3y_5 + y_4y_6 \leq 4 & 2y_{13} + y_7y_{11} + y_8y_{12} \leq 4 \\ y_{11}^2 + y_{12}^2 = 4 & 2y_{13} + y_3y_7 + y_4y_8 \leq 4 & 2y_{13} + y_9y_{11} + y_{10}y_{12} \leq 4 \\ & 2y_{13} + y_3y_9 + y_4y_{10} \leq 4 & \end{array}$$

On the above instance, the symmgroupp system reports $G_P \cong C_2 \times S_6$, generated as:

$$\langle (1, 2)(3, 4)(5, 6)(7, 8)(9, 10)(11, 12), \\ (1, 3)(2, 4), (3, 5)(4, 6), (5, 7)(6, 8), (7, 9)(8, 10), (9, 11)(10, 12) \rangle,$$

which, in original variable space, maps to:

$$\begin{aligned} & \langle (x_{11}, x_{12})(x_{21}, x_{22})(x_{31}, x_{32})(x_{41}, x_{42})(x_{51}, x_{52})(x_{61}x_{62}), \\ & (x_{11}, x_{21})(x_{12}, x_{22}), (x_{21}, x_{31})(x_{22}, x_{32}), (x_{31}, x_{41})(x_{32}, x_{42}), \\ & (x_{41}, x_{51})(x_{42}, x_{52}), (x_{51}, x_{61})(x_{52}, x_{62}) \rangle, \end{aligned}$$

or, in other words, letting $x^i = (x_{i1}, x_{i2})$ for all $i \leq 6$,

$$\langle \tau, (x^1, x^2), (x^2, x^3), (x^3, x^4), (x^4, x^5), (x^5, x^6) \rangle$$

where $\tau = \prod_{i=1}^6 (x_{i1}, x_{i2})$. Carried over to the spheres in \mathbb{R}^2 , this is a symmetric group action acting independently on the six spheres and on the two spatial dimensions.

For $N = 12, D = 3$ the formulation group is $S_3 \times S_{12}$ and for $N = 24, D = 4$ it is $S_4 \times S_{24}$. This suggests a formulation group $S_D \times S_N$ in general, where the solutions can be permuted by symmetric actions on the coordinate indices and, independently, the sphere indices. We now prove this statement formally. For all $i \leq N$ call the constraints $\|x_i\|^2 = 4$ the *center* constraints and for all $i < j \leq N$ call the constraints $\sum_{k \leq D} x_{ik}x_{jk} \leq 4 - 2\alpha$ the *distance* constraints.

THEOREM 6.1. $G_Q \cong S_D \times S_N$.

Proof. Let $(x, \alpha) \in \mathcal{G}(Q)$; the following claims are easy to establish.

1. For any $k \leq D - 1$, the permutation $\tau_k = \prod_{i \leq N} (x_{ik}, x_{i,k+1})$ is in G_Q , as both center and distance constraints are invariant w.r.t. it; notice that $\langle \tau_k \mid k \leq D - 1 \rangle \cong S_D$.
2. For any $i \leq N - 1$, the permutation $\sigma_i = \prod_{k \leq D} (x_{ik}, x_{i+1,k})$ is in G_Q , as both center and distance constraints are invariant w.r.t. it; notice that $\langle \sigma_i \mid i \leq N - 1 \rangle \cong S_N$.
3. Any permutation moving α to one of the x variables is not in G_Q . This follows because the objective function only consists of the variable α , so it is only invariant w.r.t. identity permutation.
4. For any $k \leq D - 1$, if $\pi \in G_Q$ such that $\pi(x_{ik}) = x_{i,k+1}$ for some $i \leq N$ then $\pi(x_{ik}) = x_{i,k+1}$ for all $i \leq N$, as otherwise the term $\sum_{k \leq D} x_{ik}x_{jk}$ (appearing in the distance constraints) would not be invariant.
5. For any $i \leq N - 1$, if $\pi \in G_Q$ such that $\pi(x_{ik}) = x_{i+1,k}$ for some $k \leq D$, then $\pi(x_{ik}) = x_{i+1,k}$ for all $k \leq D$, as otherwise the term $\sum_{k \leq D} x_{ik}x_{i+1,k}$ (appearing in some of the distance constraints) would not be invariant.

Let $H_D = \langle \tau_k \mid k \leq D - 1 \rangle$ and $H_N = \langle \sigma_i \mid i \leq N - 1 \rangle$. Claims 1-2 imply that $H_D, H_N \leq G_Q$. It is easy (but tedious) to check that $H_D H_N = H_N H_D$; it follows that $H_D H_N \leq G_Q$ [10] and hence H_D, H_N are normal subgroups of $H_D H_N$. Since $H_D \cap H_N = \{e\}$, we have $H_D H_N \cong H_D \times H_N \cong S_D \times S_N \leq G_Q$ [2]. Now suppose $\pi \in G_Q$ with $\pi \neq e$. By Claim 3, π cannot move α so it must map x_{ih} to x_{jk} for some $i < j \leq N, h < k \leq D$; the action $h \rightarrow k$ (resp. $i \rightarrow j$) on the components (resp. spheres) indices can be decomposed into a product of transpositions $h \rightarrow h + 1, \dots, k - 1 \rightarrow k$ (resp. $i \rightarrow i + 1, \dots, j - 1 \rightarrow j$). Thus, by Claim 4 (resp. 5), π involves a certain product γ of τ_k 's and σ_i 's; furthermore, since by definition γ maps x_{ih} to x_{jk} , any permutation in G_Q (including π) can be obtained as a product of these elements γ ; hence π is an element of $H_D H_N$, which shows $G_Q \leq H_D H_N$. Therefore, $G_Q \cong S_D \times S_N$ as claimed. \square

In problems involving Euclidean distances, it is often assumed that symmetries are rotations and translations of \mathbb{R}^n ; we remark that G_Q is not necessarily isomorphic to a (finite) subgroup of $SO(D, \mathbb{R})$. Permuting two sphere indices out of N is an action in G_Q but in general there is no rotation that can act in the same way in \mathbb{R}^D . Hence enforcing SBCs for

G_Q is not implied by simply fixing D adjacent spheres in order to break symmetries in the special orthogonal group.

By Thm. 6.1, $G_Q = \langle \tau_k, \sigma_i \mid k \leq D-1, i \leq N-1 \rangle$. It is easy to show that there is just one orbit in the natural action of G_Q on the set $A = \{1, \dots, N\} \times \{1, \dots, D\}$, and that the action of G_Q on A is not symmetric (otherwise G_Q would be isomorphic to S_{ND} , contradicting Thm. 6.1).

PROPOSITION 6.1. *For any fixed $h \leq D$,*

$$\forall i \leq N \setminus \{1\} \quad x_{i-1,h} \leq x_{ih} \quad (6.3)$$

are SBCs with respect to G_Q , $\mathcal{G}(Q)$.

Proof. Let $\bar{x} \in \mathcal{G}(Q)$; since the σ_i generate the symmetric group acting on the N spheres, there exists a permutation $\pi \in G_Q$ such that $(\bar{x}_{\pi(i),h} \mid i \leq N)$ are ordered as in (6.3). \square

6.1. Computational results on the KNP. Comparative solutions yielded by running BARON [57] on KNP instances with and without SBC reformulation have been obtained on one 2.4GHz Intel Xeon CPU of a computer with 8 GB RAM (shared by 3 other similar CPUs) running Linux. These results are shown in Table 1, which contains the following statistics at termination (occurring after 10h of user CPU time):

1. the objective function value of the incumbent
2. the seconds of user CPU time taken (meaningful if < 10h)
3. the gap still open
4. the number of BB nodes closed and those still on the tree.

The first column contains the instance name in the form `knp-N.D`. The first subsequent set of three columns refer to the solution of the original formulations (*CPU* time, best optimal objective function value f^* , open *gap* at termination, number of *nodes* created and number of open nodes in the *tree* at termination); the second subsequent set of three columns (labelled *NarrowingKNP*) refer to the solution of the formulation obtained by adjoining (6.3) to the original formulation. The last column (*R.t.*) contains the time (in user CPU seconds) needed to automatically compute the formulation group using the methods in Sect. 4. In both formulations we fixed the first sphere at $(-2, 0, \dots, 0)$ to break some of the orthogonal symmetries. We remark that the objective function values are negative because we are using a minimization direction (instead of maximization).

Judging from the 2-dimensional KNP instances, where BARON converges to optimality, it is evident that the *NarrowingKNP* reformulation is crucial to decrease the CPU time significantly: the total CPU time needed to solve the five 2D instances in the original formulation is 74047s, whereas the *NarrowingKNP* reformulations only take 173s, that is a gain of over 400 times. It also appears clear from the results relating to the larger instances that adjoining SBCs to the formulation makes a definite (positive) difference in the exploration rate of the search tree. The beneficial effects of the narrowing decrease with the instance size (to the extent of disappearing

TABLE 1
Computational results for the Kissing Number Problem.

Instance	Slv	Original problem			NarrowingKNP			R.t.
		CPU	f^* gap	nodes tree	CPU	f^* gap	nodes tree	
knp-6_2	B	8.66	-1 0%	1118 0	1.91	-1 0%	186 0	1.43
knp-7_2	B	147.21	-0.753 0%	13729 0	3.86	-0.753 0%	260 0	1.47
knp-8_2	B	1892	-0.586 0%	179994 0	12.17	-0.586 0%	650 0	2.94
knp-9_2	B	36000	-0.47 33.75%	1502116 176357	37.36	-0.47 0%	1554 0	1.96
knp-10_2	B	36000	-0.382 170%	936911 167182	117.79	-0.382 0%	3446 0	1.97
knp-12_3	B	36000	-1.105 8.55%	299241 12840	36000	-1.105 8.55%	273923 5356	3.39
knp-13_3	B	36000	-0.914 118%	102150 64499	36000	-0.914 118%	68248 33013	3.38
knp-24_4	B	36000	-0.966 107%	10156 7487	36000	-0.92 117%	4059 2985	5.62
knp-24_5	B	36000	-0.93 116%	7768 5655	36000	-0.89 124%	4251 3122	6.1

completely for knp-25_4) because we are keeping the CPU time fixed at 10h. We remark that the effectiveness of the *NarrowingKNP* reformulation in low-dimensional spaces can be partly explained by the fact that it is designed to break sphere-related symmetries rather than dimension-related ones (naturally, the instance size also counts: the largest 2D instance, knp-10_2, has 21 variables, whereas the smallest 3D one, knp-12_3, has 37 variables).

7. Conclusion. This paper introduces the study of symmetries in nonlinear and mixed-integer nonlinear programming. We use a generalization of the definition of formulation group given by Margot, based on transforming a mathematical programming formulation into a DAG. This allows automatic symmetry detection using graph isomorphism tools. Symmetries are then broken by means of static symmetry-breaking inequalities. We present an application of our findings to the Kissing Number Problem.

Acknowledgements. I wish to thank François Margot for many useful discussions and suggestions, and for carefully checking [34] (from which some of the present material is taken) as well as one particularly careful referee. This work was financially supported by grants: ANR 07-JCJC-0151 “ARS”, Digiteo Chair 2009-14D “RMNCCO”, Digiteo Emergence 2009-55D “ARM”.

REFERENCES

- [1] F. ALIZADEH. Interior point methods in Semidefinite Programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, **5**(1):13–51, 1995.

- [2] R. ALLENBY. *Rings, Fields and Groups: an Introduction to Abstract Algebra*. Edward Arnold, London, 1991.
- [3] L. BABAI. Automorphism groups, isomorphism, reconstruction. In R. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, Vol. 2, pages 1447–1540. MIT Press, Cambridge, MA, 1996.
- [4] C. BACHOC AND F. VALLENTIN. New upper bounds for kissing numbers from Semidefinite Programming. *Journal of the American Mathematical Society*, **21**:909–924, 2008.
- [5] D. BELL. Constructive group relaxations for integer programs. *SIAM Journal on Applied Mathematics*, **30**(4):708–719, 1976.
- [6] P. BELOTTI, J. LEE, L. LIBERTI, F. MARGOT, AND A. WÄCHTER. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, **24**(4):597–634, 2009.
- [7] T. BERTHOLD AND M. PFETSCH. Detecting orbitopal symmetries. In B. Fleischmann, K.-H. Borgwardt, R. Klein, and A. Tuma, editors, *Operations Research Proceedings 2008*, pages 433–438, Berlin, 2009. Springer.
- [8] K. BOOTH AND C. COLBOURN. Problems polynomially equivalent to graph isomorphism. Technical Report CS-77-04, University of Waterloo, 1979.
- [9] M. BOULLE. Compact mathematical formulation for graph partitioning. *Optimization and Engineering*, **5**:315–333, 2004.
- [10] A. CLARK. *Elements of Abstract Algebra*. Dover, New York, 1984.
- [11] D. COHEN, P. JEAVONS, C. JEFFERSON, K. PETRIE, AND B. SMITH. Symmetry definitions for constraint satisfaction problems. In P. van Beek, editor, *Constraint Programming*, Vol. **3709** of *LNCS*. Springer, 2005.
- [12] A. COSTA, P. HANSEN, AND L. LIBERTI. Formulation symmetries in circle packing. In R. Mahjoub, editor, *Proceedings of the International Symposium on Combinatorial Optimization*, Vol. **36** of *Electronic Notes in Discrete Mathematics*, pages 1303–1310, Amsterdam, 2010. Elsevier.
- [13] J. CRAWFORD, M. GINSBERG, E. LUKS, AND A. ROY. Symmetry-breaking predicates for search problems. In *Principles of Knowledge Representation and Reasoning*, pages 148–159, Cambridge, MA, 1996. Morgan Kaufmann.
- [14] PH. DELSARTE. Bounds for unrestricted codes by linear programming. *Philips Research Reports*, **27**:272–289, 1972.
- [15] Y. FAENZA AND V. KAIBEL. Extended formulations for packing and partitioning orbitopes. *Mathematics of Operations Research*, **34**(3):686–697, 2009.
- [16] R. FOURER AND D. GAY. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.
- [17] E.J. FRIEDMAN. Fundamental domains for integer programs with symmetries. In A. Dress, Y. Xu, and B. Zhu, editors, *COCOA Proceedings*, Vol. **4616** of *LNCS*, pages 146–153. Springer, 2007.
- [18] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4.10*, 2007.
- [19] K. GATERMANN AND P. PARRILLO. Symmetry groups, Semidefinite Programs and sums of squares. *Journal of Pure and Applied Algebra*, **192**:95–128, 2004.
- [20] R. GOMORY. Some polyhedra related to combinatorial problems. *Linear Algebra and Its Applications*, **2**(4):451–558, 1969.
- [21] M. HALL. *Theory of Groups*. Chelsea Publishing Company, New York, 2nd edition, 1976.
- [22] P. HANSEN AND N. MLAĐENOVIC. Variable neighbourhood search: Principles and applications. *European Journal of Operations Research*, **130**:449–467, 2001.
- [23] K. HERR AND R. BÖDI. Symmetries in linear and integer programs. Technical Report 0908.3329v1 [math.CO], arXiv.org, 2009.
- [24] ILOG. *ILOG CPLEX 11.0 User’s Manual*. ILOG S.A., Gentilly, France, 2008.
- [25] E. JOHNSON. *Integer Programming: Facets, Subadditivity and Duality for Group and Semi-group Problems*. SIAM, Philadelphia, 1980.
- [26] V. KAIBEL AND M. PFETSCH. Packing and partitioning orbitopes. *Mathematical Programming*, **114**(1):1–36, 2008.

- [27] Y. KANNO, M. OHSAKI, K. MUROTA, AND N. KATOH. Group symmetry in interior-point methods for Semidefinite Program. *Optimization and Engineering*, **2**:293–320, 2001.
- [28] E. DE KLERK AND R. SOTIROV. Exploiting group symmetry in Semidefinite Programming relaxations of the quadratic assignment problem. *Mathematical Programming*, **122**(2):225–246, 2010.
- [29] D.E. KNUTH. *The Art of Computer Programming, Part I: Fundamental Algorithms*. Addison-Wesley, Reading, MA, 1968.
- [30] S. KUCHERENKO, P. BELOTTI, L. LIBERTI, AND N. MACULAN. New formulations for the kissing number problem. *Discrete Applied Mathematics*, **155**(14):1837–1841, 2007.
- [31] J. LEE AND F. MARGOT. On a binary-encoded ILP coloring formulation. *INFORMS Journal on Computing*, **19**(3):406–415, 2007.
- [32] L. LIBERTI. Writing global optimization software. In L. Liberti and N. Maculan, editors, *Global Optimization: from Theory to Implementation*, pages 211–262. Springer, Berlin, 2006.
- [33] L. LIBERTI. Automatic generation of symmetry-breaking constraints. In B. Yang, D.-Z. Du, and C.A. Wang, editors, *COCOA Proceedings*, Vol. **5165** of *LNCS*, pages 328–338, Berlin, 2008. Springer.
- [34] L. LIBERTI. Reformulations in mathematical programming: Symmetry. Technical Report 2165, Optimization Online, 2008.
- [35] L. LIBERTI. Reformulations in mathematical programming: Definitions and systematics. *RAIRO-RO*, **43**(1):55–86, 2009.
- [36] L. LIBERTI. Reformulations in mathematical programming: Automatic symmetry detection and exploitation. *Mathematical Programming*, DOI 10.1007/s10107-010-0351-0.
- [37] L. LIBERTI, S. CAFIERI, AND F. TARISSAN. Reformulations in mathematical programming: A computational approach. In A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht, editors, *Foundations of Computational Intelligence* Vol. **3**, number 203 in Studies in Computational Intelligence, pages 153–234. Springer, Berlin, 2009.
- [38] L. LIBERTI, N. MLAĐENOVIC, AND G. NANNICINI. A good recipe for solving MINLPs. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Hybridizing metaheuristics and mathematical programming*, Vol. **10** of *Annals of Information Systems*, pages 231–244, New York, 2009. Springer.
- [39] F. MARGOT. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, **94**:71–90, 2002.
- [40] F. MARGOT. Exploiting orbits in symmetric ILP. *Mathematical Programming B*, **98**:3–21, 2003.
- [41] F. MARGOT. Small covering designs by branch-and-cut. *Mathematical Programming B*, **94**:207–220, 2003.
- [42] F. MARGOT. Symmetric ILP: coloring and small integers. *Discrete Optimization*, **4**:40–62, 2007.
- [43] F. MARGOT. Symmetry in integer linear programming. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, editors, *50 Years of Integer Programming*, pages 647–681. Springer, Berlin, 2010.
- [44] B. MCKAY. Practical graph isomorphism. *Congressus Numerantium*, **30**:45–87, 1981.
- [45] B. MCKAY. *nauty User's Guide (Version 2.4)*. Computer Science Dept., Australian National University, 2007.
- [46] C. MEARS, M. GARCIA DE LA BANDA, AND M. WALLACE. On implementing symmetry detection. *Constraints*, **14**(2009):443–477, 2009.

- [47] C. MEARS, M. GARCIA DE LA BANDA, M. WALLACE, AND B. DEMOEN. A novel approach for detecting symmetries in CSP models. In L. Perron and M. Trick, editors, *Constraint Programming, Artificial Intelligence and Operations Research*, volume **5015** of *LNCS*, pages 158–172, New York, 2008. Springer.
- [48] O. MUSIN. The kissing number in four dimensions. *arXiv:math.MG/0309430v2*, April 2005.
- [49] A. NEUMAIER. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, **13**:271–369, 2004.
- [50] J. OSTROWSKI, J. LINDEROTH, F. ROSSI, AND S. SMRIGLIO. Orbital branching. In M. Fischetti and D.P. Williamson, editors, *IPCO*, volume **4513** of *LNCS*, pages 104–118. Springer, 2007.
- [51] J. OSTROWSKI, J. LINDEROTH, F. ROSSI, AND S. SMRIGLIO. Constraint orbital branching. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *IPCO*, volume **5035** of *LNCS*, pages 225–239. Springer, 2008.
- [52] J.-F. PUGET. Automatic detection of variable and value symmetries. In P. van Beek, editor, *Constraint Programming*, volume **3709** of *LNCS*, pages 475–489, New York, 2005. Springer.
- [53] A. RAMANI AND I. MARKOV. Automatically exploiting symmetries in constraint programming. In B. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *Constraint Solving and Constraint Logic Programming*, volume **3419** of *LNAI*, pages 98–112, Berlin, 2005. Springer.
- [54] S. ROBERTSON. *Polytopes and Symmetry*. Cambridge University Press, Cambridge, 1984.
- [55] R.T. ROCKAFELLAR. A combinatorial algorithm for linear programs in the general mixed form. *Journal of the Society for Industrial and Applied Mathematics*, **12**(1):215–225, 1964.
- [56] K.H. ROSEN, editor. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, New York, 2000.
- [57] N.V. SAHINIDIS AND M. TAWARMALANI. *BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs*, User’s Manual, 2005.
- [58] H. SCHICHL AND A. NEUMAIER. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, **33**(4):541–562, 2005.
- [59] K. SCHÜTTE AND B.L. VAN DER WAERDEN. Das problem der dreizehn kugeln. *Mathematische Annalen*, **125**:325–334, 1953.
- [60] A. SERESS. *Permutation Group Algorithms*. Cambridge University Press, Cambridge, 2003.
- [61] H. SHERALI AND C. SMITH. Improving discrete model representations via symmetry considerations. *Management Science*, **47**(10):1396–1407, 2001.
- [62] E. SMITH AND C. PANTELIDES. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, **23**:457–478, 1999.
- [63] G. SZPIRO. Newton and the kissing problem. *Plus magazine (online)*, **23**, January 2003.
- [64] A.W. TUCKER. A combinatorial equivalence of matrices. In R. Bellman and M. Hall, editors, *Proceedings of the 10th Symposium of Applied Mathematics*, pages 129–140, Providence, Rhode Island, 1960. AMS.
- [65] A.W. TUCKER. Solving a matrix game by linear programming. *IBM Journal of Research and Development*, **4**:507–517, 1960.
- [66] A.W. TUCKER. Combinatorial theory underlying linear programs. In L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*. McGraw-Hill, New York, 1963.
- [67] R. UEHARA, S. TODA, AND T. NAGOYA. Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. *Discrete Applied Mathematics*, **145**:479–482, 2005.
- [68] F. VALLENTIN. Symmetry in Semidefinite Programs. *Linear Algebra and its Applications*, **430**:360–369, 2009.

- [69] L. WOLSEY. Group representation theory in integer programming. Technical Report Op. Res. Center 41, MIT, 1969.
- [70] W. ZHU. Unsolvability of some optimization problems. *Applied Mathematics and Computation*, **174**:921–926, 2006.

PART V:

CONVEXIFICATION AND LINEARIZATION

USING PIECEWISE LINEAR FUNCTIONS FOR SOLVING MINLPs

BJÖRN GEISLER*, ALEXANDER MARTIN*, ANTONIO MORSI*, AND LARS SCHEWE*

1. Introduction. In this chapter we want to demonstrate that in certain cases general mixed integer nonlinear programs (MINLPs) can be solved by just applying purely techniques from the mixed integer linear world. The way to achieve this is to approximate the nonlinearities by piecewise linear functions. The advantage of applying mixed integer linear techniques are that these methods are nowadays very mature, that is, they are fast, robust, and are able to solve problems with up to millions of variables. In addition, these methods have the potential of finding globally optimal solutions or at least to provide solution guarantees. On the other hand, one tends to say at this point “If you have a hammer, everything is a nail.”[15], because one tries to reformulate or to approximate an actual nonlinear problem until one obtains a model that is tractable by the methods one is common with. Besides the fact that this is a very typical approach in mathematics the question stays whether this is a reasonable approach for the solution of MINLPs or whether the nature of the nonlinearities inherent to the problem gets lost and the solutions obtained from the mixed integer linear problem have no meaning for the MINLP. The purpose of this chapter is to discuss this question. We will see that the truth lies somewhere in between and that there are problems where this is indeed a reasonable way to go and others where it is not.

The idea to obtain a mixed integer linear program (MIP) out of a MINLP is simple. Each nonlinear function is approximated by a piecewise linear function. This reads for 1-dimensional functions as follows: We subdivide the interval $[a, b]$, where we want to approximate the nonlinear function $f(x)$ by introducing vertices $[a = \bar{x}_0 < \bar{x}_2 < \dots < \bar{x}_n = b]$ and determine the function value $\bar{y}_i = f(\bar{x}_i)$ at each of these vertices. The pairs $(\bar{x}_{i-1}, \bar{y}_{i-1})$ and (\bar{x}_i, \bar{y}_i) might then be connected by lines for $i = 1, \dots, n$ to obtain a piecewise linear function. In higher dimensions this principle is the same except that the lines are triangles, tetrahedra, or in general simplices. This fact immediately shows one major problem with this approach. The number of simplices needed increases drastically with the dimension, for instance, even the subdivision of a cube in dimension five contains at least 25 simplices, and in dimension six already 86. Thus this approach seems applicable for MINLPs only if the nonlinear functions contain only a few variables. We will see such examples later on.

*Department of Mathematics, Friedrich-Alexander-University of Erlangen-Nuremberg, Erlangen, Germany ({[bjoern.geissler](mailto:bjoern.geissler@math.uni-erlangen.de), [alexander.martin](mailto:alexander.martin@math.uni-erlangen.de), [antonio.morsi](mailto:antonio.morsi@math.uni-erlangen.de), [lars.schewe](mailto:lars.schewe@math.uni-erlangen.de)}@math.uni-erlangen.de).

On the other hand this drawback might be overcome by modeling high dimensional nonlinear functions by sums of one-dimensional functions by using the logarithm and by substituting variables. For instance the nonlinear equation $f = xy$ with $x, y \in \mathbb{R}_+$ may be modeled by substituting $u = \log(x)$, $v = \log(y)$ and $f' = \log(f)$. Now we can rewrite the nonlinear equation in two variables by the linear equation $f' = u + v$ consisting of sums of only 1-dimensional functions. Functions in this form are called separable and are discussed in Section 2. However, we already see at this stage that these transformations might cause numerical difficulties due to the application of the log.

If it is not advisory to apply these transformations or if the nonlinear function is not separable we have to approximate the nonlinear functions as they are by piecewise linear functions. And in fact the literature is huge on how to model piecewise linear functions. Starting in the fifties where it was suggested to model piecewise linear functions by convex combinations of the vertices various new ideas have come up until today. A paper by Nemhauser and Vielma [23] marks the current end. All these methods will be discussed in Section 3. Interesting to note is that from a theoretical point of view some of the methods are clearly superior to others, but this fact cannot yet be confirmed by numerical results as we will see in Section 6.

What we have left out so far is a discussion about the introduction of the vertices in order to get a good piecewise linear approximation. On one side, the approach has the potential to approximate the nonlinear functions arbitrarily close by introducing enough vertices. On the other hand, the introduction of additional vertices implies the introduction of additional binary variables or combinatorial conditions, and thus one must be careful in order not to end up in intractable models. Thus, an interesting side problem occurs how to introduce not too many of such vertices keeping simultaneously the approximation error under control. This topic is addressed in Section 4.

Before starting the discussions in detail let us introduce some motivating practical examples, where piecewise linear functions seem to be a reasonable way to go. As pointed out above, the piecewise linear approach for the solution of MINLPs seems to be appropriate when the involved nonlinear functions are of low dimension. This is true for example for any kind of network problems with nonlinearities occurring on the edges such as the design and management of energy networks. For instance, in gas network optimization, the transport of gas is modeled by a system of partial differential equations. After discretization of the PDEs a system of differential algebraic equations is left, where the nonlinearities depend on the ingoing pressure, the outgoing pressure and the flow of the gas in each pipe. The PDE or DAE systems, respectively, model the pressure loss of the gas in the pipes. Similar, the fuel gas consumptions of the compressors, which are used to increase the pressure of the gas again, depend also on these three

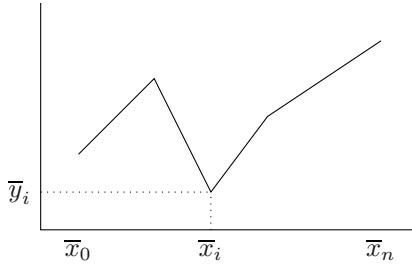
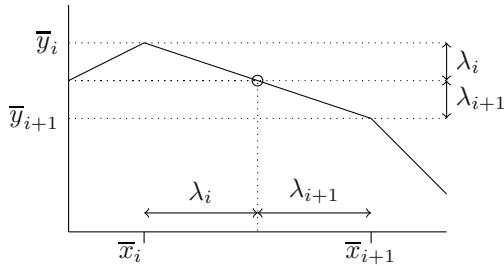
types of variables, see Section 6 for details. The same type of nonlinearities occur when one wants to model water or power networks.

Despite the facts that piecewise linear functions might be helpful for the solution of MINLPs and that they are of interest of their own, they also directly show up in practical applications. One such example is the optimization of the social welfare in power spot markets. One of the products that is traded for instance at the European Energy Exchange (EEX) or the Nordpol Spot (NPS) are hourly bid curves. For each trading hour of the day the customers give their hourly bids by a set of points of the form power per price. These set of points are linearly interpolated resulting in a piecewise linear function. All other conditions in the welfare optimization problem are of linear or combinatorial nature resulting in a huge mixed integer linear program containing piecewise linear functions, see [16] for details.

2. Linearizing 1D- and separable functions. We start our discussion with the case of univariate functions. The methods can directly be adapted to the case of separable functions. Additionally, most methods for higher dimensional functions are extensions of the one-dimensional case, so the understanding of this easier case is helpful for the study of the more complicated methods. However, we have already seen in the introduction that also one-dimensional piecewise linear functions occur in applications. We will assume in this section that we are already dealing with a piecewise linear function, for methods how to discretize given nonlinear functions we refer to Section 4.

The case of separable functions can be reduced to the case of one-dimensional functions. A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is called *separable*, if it can be written as a sum of one-dimensional functions, i.e., there exist $f_1, \dots, f_k : \mathbb{R} \rightarrow \mathbb{R}$ such that $f(x_1, \dots, x_k) = \sum_{i=1}^k f_i(x_i)$. However, one should note that in many cases where a function is given analytically it can be transformed into a composition of separable functions. For instance a monomial function $f(x) = \prod_{i=1}^d x_i^{\alpha_i}$ can be transformed into $g(x) = \sum_{i=1}^d \alpha_i \log x_i$ so that $f = \exp \circ g$.

For the rest of this section we will study the following situation (see Fig 1): We are given a continuous piecewise linear function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ that is encoded by $n + 1$ vertices \bar{x}_i ($i \in \{0, \dots, n\}$) and the respective function values $\bar{y}_i = \phi(\bar{x}_i)$. Thus, the function consists of n line-segments. The aim of this section is to formulate a mixed integer linear model of the relation $y = \phi(x)$ for a point $x \in [\bar{x}_0, \bar{x}_n]$. For this we introduce auxiliary variables to model the piecewise linear geometry. However, in all models it is necessary to enforce some combinatorial constraints on these auxiliary variables. For this we use additional binary variables, these will be called z (with respective subscripts) throughout the chapter.

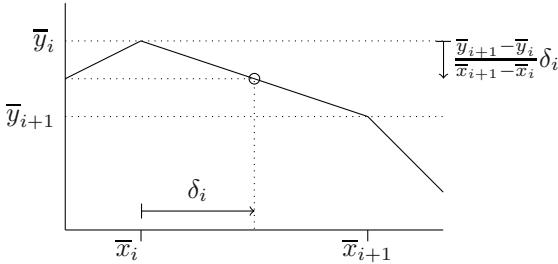
FIG. 1. *A piecewise linear function.*FIG. 2. *Convex-combination method.*

We start with the *convex-combination method* [7] (see Fig. 2). It uses the following observation: When ϕ is a piecewise linear function, we can compute the function value at point x if we can express x as the convex combination of the neighboring nodes. We know from Carathéodory's theorem that we only need these two neighboring points. This condition can be expressed using n binary variables z_1, \dots, z_n . Thus, we use the following model

$$x = \sum_{i=0}^n \lambda_i \bar{x}_i, \quad \sum_{i=0}^n \lambda_i = 1, \quad \lambda \geq 0 \quad (2.1)$$

$$y = \sum_{i=0}^n \lambda_i \bar{y}_i, \quad (2.2)$$

$$\begin{aligned} & \lambda_0 \leq z_1, \\ & \forall i \in \{1, \dots, n-1\}. \quad \lambda_i \leq z_i + z_{i+1}, \\ & \lambda_n \leq z_n \\ & \sum_{i=1}^n z_i \leq 1. \end{aligned} \quad (2.3)$$

FIG. 3. *Incremental method.*

Another method to model a piecewise linear function is the *incremental method* [21] (see Fig. 3). Here we express x lying in interval i as $\bar{x}_{i-1} + \delta$. The function value can then be expressed as $y = \bar{y}_{i-1} + \frac{\bar{y}_i - \bar{y}_{i-1}}{\bar{x}_i - \bar{x}_{i-1}} \delta$. To use this in a formulation we use variables δ_i for each interval i such that $0 \leq \delta_i \leq \bar{x}_i - \bar{x}_{i-1}$. We then use binary variables z_i to force the so-called “filling condition”, i.e., the condition $\delta_i > 0$ implies that δ_{i-1} is at its upper bound.

We obtain the following model

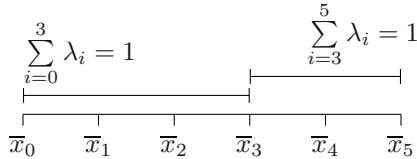
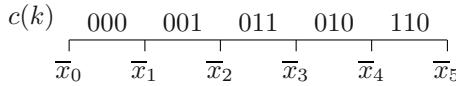
$$x = \bar{x}_0 + \sum_{i=1}^n \delta_i \quad (2.4)$$

$$y = \bar{y}_0 + \sum_{i=1}^n \frac{\bar{y}_i - \bar{y}_{i-1}}{\bar{x}_i - \bar{x}_{i-1}} \delta_i \quad (2.5)$$

$$\begin{aligned} \forall i \in \{1, \dots, n\}. (\bar{x}_{i-1} - \bar{x}_{i-2}) z_{i-1} &\leq \delta_i \\ \forall i \in \{1, \dots, n-1\}. \delta_i &\leq (\bar{x}_i - \bar{x}_{i-1}) z_i. \end{aligned} \quad (2.6)$$

The quality of these two schemes was discussed by Padberg [24]. He studied the case where the objective function of a linear program is a piecewise linear function. In that case the formulation of the incremental method always yields an integral solution of the LP-relaxation whereas this is not the case for the convex combination method. In any case the polyhedron described by the incremental method is properly contained of the one described by the convex combination method. These are not the only methods that were proposed to model piecewise linear functions: In particular we want to mention the so-called *multiple choice method* [1].

The methods described so far are generic and be readily incorporated in standard mixed integer linear models. However, the number of newly introduced binary variables often slows down the solver. In addition, branching on the binary variables tends to result in deep and unbalanced branch-and-bound trees. This problem was noticed quite early in the development

FIG. 4. *Branching decision for SOS-2 constraints.*FIG. 5. *Gray code encoding for SOS-2 constraints.*

of these methods. The proposed remedy was to modify the branching strategy of the solver (see e.g. [3, 4]). Thus, to model the combinatorial constraints of the respective models we no longer use the auxiliary binary variables, but we incorporate the constraints directly in the branching. We describe these modifications for the case of the convex combination method; in the case of the incremental formulation a similar approach can be made (cf. [13]).

Looking at the convex combination method we see that we need to enforce condition (2.3) on the components of λ : At most two components of λ may be non-zero, and if two components are non-zero, these have to be adjacent. This condition is called a Special Ordered Set Condition of Type 2 (SOS-2). Thus, if the LP-relaxation yields some λ that violates these conditions one can define new branches in the branch-and-bound-tree by generating a node with the additional constraint that $\sum_{i=0}^k \lambda_i = 1$

and another one with the additional constraint $\sum_{i=k}^n \lambda_i = 1$ (see Fig. 4). One can summarize this by saying that we take the decision in which of the n intervals the point x should lie. As mentioned earlier, the main drawback of the classical formulations with binary variables is the unbalanced branch-and-bound tree that we obtain. On the other hand, the interaction between the binary variables from the formulation of the piecewise linear function with the rest of the model may yield speed ups compared to an approach with modified branching.

However, it has been pointed out recently by Vielma and Nemhauser in [23] that one can formulate conditions like the SOS-2 condition using dramatically fewer binary variables. We discuss their formulation for the SOS-2 case, i.e., the convex-combination method. We have already seen that with n binary variables we can encode 2^n different cases, so we should

be able to model an SOS-2 condition with only $\lceil \log_2 n \rceil$ binary variables. And that is indeed the case.

The idea is to use a Gray code to encode the definition intervals of the piecewise linear function. We take an injective function $c : \{1, \dots, n\} \rightarrow \{0, 1\}^{\lceil \log_2 n \rceil}$ with the additional property that for any number k the vectors $c(k)$ and $c(k+1)$ only differ in one component. For an example see Fig. 5. Using binary variables $z_1, \dots, z_{\lceil \log_2 n \rceil}$, we can then enforce the SOS-2 condition with the following constraints (note that we still need the constraints (2.1)):

$$\forall k \in \{1, \dots, n\} \sum_{i=0}^{k-2} \lambda_i + \sum_{i=k+1}^n \lambda_i \leq \sum_{\{l | (c(k))_l = 1\}} (1 - z_l) + \sum_{\{l | (c(k))_l = 0\}} z_l. \quad (2.7)$$

This approach can even further be generalized, see Nemhauser and Vielma [23] and Vielma et al. [30]. The model we present here can be viewed as the one-dimensional variant of the logarithmic formulation of the aggregated convex combination method (again see [30]). The so-called disaggregated formulation will be presented in the next section in the context of nonseparable functions.

In general, the methods here can be adapted to special cases. For instance, sometimes one may drop the SOS-2 constraints for the convex combination method. This is the case for convex ϕ , if it is the objective function or it only appears in a constraint of the form $\phi(x) \leq b$. In addition, the mentioned methods can also be extended to the case of discontinuous ϕ , see [8, 31, 30].

3. Piecewise linearization of nonseparable functions. In this section we present adequate mixed integer modeling techniques for higher dimensional functions. Even if the methods from the previous section are not restricted to univariate functions, as we have seen with separable functions, many functions do not fall into these category. Commonly in that case modeling tricks can be applied to reduce the degree of a nonlinear term, but this tricks often cause numerically unstable programs.

The univariate MIP approaches from the previous section provide a basis for the following models of multivariate piecewise linear functions. We review the presented approaches with a view to piecewise linear functions of arbitrary dimension.

For the remainder of this section let $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ be an arbitrary continuous piecewise linear function. Thus the function ϕ does not necessarily have to be separable and of course it does not need to be convex and concave, respectively. First, we generalize the definition of piecewise linear functions to higher dimensions.

DEFINITION 3.1. Let $\mathcal{D} \subset \mathbb{R}^d$ be a compact set. A continuous function $\phi : \mathcal{D} \rightarrow \mathbb{R}$ is called piecewise linear if it can be written in the form

$$\phi(x) = \phi_S(x) \quad \text{for } x \in S \quad \forall S \in \mathcal{S}, \quad (3.1)$$

with affine functions ϕ_S for a finite set of simplices \mathcal{S} that partitions \mathcal{D} .

For the sake of simplicity we restrict ourselves to functions over compact domains, though some techniques can be applied to unbounded domains, too. Furthermore our MIP techniques deal with continuous functions only. More information on a special class of discontinuous functions so-called lower semi-continuous functions can be found in [31]. In the literature it is common to be not as restrictive and require the domain S of each piece to be simply a polytope. However, since both definitions are equivalent and some of our approaches rely on simplicial pieces we go for the above definition.

According to Definition 3.1, we denote by \mathcal{S} the set of simplices forming \mathcal{D} . The cardinality of this set is $n := |\mathcal{S}|$. The set of vertices of a single d -simplex S is denoted by $\mathcal{V}(S) := \{\bar{x}_0^S, \dots, \bar{x}_d^S\}$. Furthermore $\mathcal{V}(\mathcal{S}) = \{\bar{x}_1^S, \dots, \bar{x}_m^S\} := \bigcup_{S \in \mathcal{S}} \mathcal{V}(S)$ is the entire set of vertices of \mathcal{S} . As in the previous section our aim is to formulate a mixed integer linear model in which $y = \phi(x)$ for $x \in \mathcal{D}$ holds. According to the univariate case auxiliary variables will uniformly be denoted by z .

We start by adapting the *convex-combination method* to d -dimensional functions. The key idea here is that every point $x \in S \subseteq \mathcal{D}$ can be expressed as convex combination of vertices of the simplex S . In addition with binary variables z_1, \dots, z_n used to decide which simplex contains the point x we yield the following model.

$$x = \sum_{j=1}^m \lambda_j \bar{x}_j^S, \sum_{j=1}^m \lambda_j = 1, \lambda \geq 0, \quad (3.2)$$

$$y = \sum_{j=1}^m \lambda_j \bar{y}_j^S, \quad (3.3)$$

$$\lambda_j \leq \sum_{\{i \mid \bar{x}_j^S \in \mathcal{V}(S_i)\}} z_i \quad \text{for } j = 1, \dots, m, \quad (3.4)$$

$$\sum_{i=1}^n z_i \leq 1. \quad (3.5)$$

A slight modification of this model is known as *disaggregated convex combination method* [22]. Its name results from explicitly introducing λ -variables for each simplex' vertex. Originally, this approach is designed

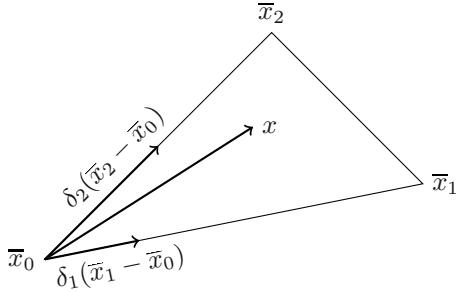


FIG. 6. A point $x = \bar{x}_0 + \delta_1(\bar{x}_1 - \bar{x}_0) + \delta_2(\bar{x}_2 - \bar{x}_0)$ inside a triangle.

for semi-continuous functions, but on some continuous model instances the drawback of much more variables is evened by the fact that this formulation, unlike the aggregated variant, is locally ideal. The term of locally ideal MIP formulations for piecewise linear functions has been established in [24, 25] and states that the polytope of the corresponding LP relaxation, without further constraints, is integral.

Adapting the *incremental method* to higher dimensional functions is not as simple as the convex combination method as we will notice. At first we see that any point x^S inside a simplex $S \in \mathcal{S}$ can be expressed either as convex combination of its vertices or equivalently as $x^S = \bar{x}_0^S + \sum_{j=1}^d (\bar{x}_j^S - \bar{x}_0^S) \delta_j^S$ with $\sum_{j=1}^d \delta_j^S \leq 1$ and nonnegative $\delta_i^S \geq 0$ for $i = 1, \dots, d$ (cf. Fig. 6).

When we look back to the incremental method in dimension one, we notice that the second main argument of this approach is that an ordering of the simplices holds in which the last vertex of any simplex is equal to the first vertex of the next one. A natural generalization of this approach to dimension $d \geq 2$ is therefore possible if an ordering of simplices with the following properties is available.

- (O1) The simplices in $\mathcal{S} = \{S_1, \dots, S_n\}$ are ordered in such a way that $S_i \cap S_{i+1} \neq \emptyset$ for $i = 1, \dots, n-1$ holds and
- (O2) for each simplex S_i its vertices $\bar{x}_0^{S_i}, \dots, \bar{x}_d^{S_i}$ can be labeled such that $\bar{x}_d^{S_i} = \bar{x}_0^{S_{i+1}}$ holds for $i = 1, \dots, n-1$.

An ordering of a two-dimensional example triangulation is illustrated in Fig. 7. In the univariate case such an ordering is automatically given since the set of simplices \mathcal{S} simply consists of a sequence of line segments. Based on properties (O1) and (O2) the first vertex $\bar{x}_0^{S_i}$ of simplex S_i is obtained by $\bar{x}_0^{S_i} = \bar{x}_0^{S_1} + \sum_{k=1}^{i-1} (\bar{x}_d^{S_k} - \bar{x}_0^{S_k})$.

Bringing this together with the representation of a point inside a single simplex as sum of a vertex and the rays spanning the simplex from that vertex, we get the generalized incremental model

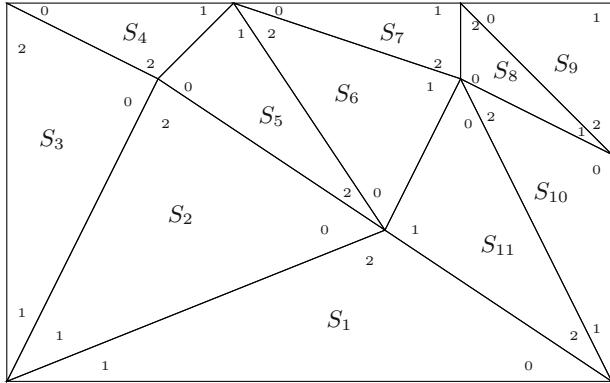


FIG. 7. Triangle ordering on a rectangular domain.

$$x = \bar{x}_0^{S_1} + \sum_{i=1}^n \sum_{j=1}^d (\bar{x}_j^{S_i} - \bar{x}_0^{S_i}) \delta_j^{S_i}, \quad (3.6)$$

$$y = \bar{y}_0^{S_1} + \sum_{i=1}^n \sum_{j=1}^d (\bar{y}_j^{S_i} - \bar{y}_0^{S_i}) \delta_j^{S_i}, \quad (3.7)$$

$$\sum_{j=1}^d \delta_j^{S_i} \leq 1 \quad \text{for } i = 1, \dots, n, \quad (3.8)$$

$$\delta_j^{S_i} \geq 0 \quad \text{for } i = 1, \dots, n \text{ and } j = 1, \dots, d. \quad (3.9)$$

In addition the δ -variables have to satisfy a “generalized filling condition”, that is, if for any simplex S_i a variable $\delta_j^{S_i}$ is positive then $\delta_d^{S_{i-1}} = 1$ must hold. Obviously we can conclude that a variable $\delta_j^{S_i}$ can only be positive if for all previous simplices $k = 1, \dots, i-1$ the variables $\delta_d^{S_k}$ are equal to one. To enforce this condition we introduce auxiliary binary variables z and use the constraints

$$\sum_{j=1}^d \delta_j^{S_{i+1}} \leq z_i \quad \text{for } i = 1, \dots, n-1, \quad (3.10)$$

$$z_i \leq \delta_d^{S_i} \quad \text{for } i = 1, \dots, n-1. \quad (3.11)$$

Likewise the univariate case, the integrality of the polytope described by inequalities (3.8)-(3.11) together with the nonnegativity constraints $z_i \geq 0$ for $i = 1, \dots, n - 1$, is guaranteed as shown by Wilson [32].

In the following we will characterize for which simplicial sets an ordering with properties (O1) and (O2) exists as well as how this ordering can be computed efficiently. In case of a univariate function, ordering the simplices according to (O1) and (O2) is trivially done by labeling the line segment connecting $(\bar{x}_{i-1}, \bar{y}_{i-1})$ and (\bar{x}_i, \bar{y}_i) with S_i for $i = 2, \dots, n$. For bivariate piecewise linear functions the simplices are triangles and it is somewhat more involved to determine an appropriate labeling scheme. As early as in 1979 Todd introduced an ordering of the simplices for triangulations of type K_1 and J_1 [29], before Wilson [32] showed that the adjacency graph of any triangulation of a domain that is a topological disc in \mathbb{R}^2 is hamiltonian. Here, the adjacency graph is constructed, by introducing a vertex for each simplex of the triangulation and an edge between any two vertices, whose corresponding simplices have at least one common point. Eventually, Bartholdi and Goldsman gave an algorithm with running time $\mathcal{O}(n^2)$ to calculate a hamiltonian path through the adjacency graph of any facet-adjacent triangulation in \mathbb{R}^2 [2]. We call a triangulation with d -simplices facet-adjacent, if for each nonempty subset $\mathcal{S}' \subset \mathcal{S}$ there exist d -simplices $S' \in \mathcal{S}'$ and $S \in \mathcal{S} \setminus \mathcal{S}'$, such that $\mathcal{V}(S') \cap \mathcal{V}(S) = d$. In other words, a triangulation is facet-adjacent, if and only if the graph consisting of the vertices and edges of the triangulation is d -connected. For example, every triangulation of a convex set is facet-adjacent. For $d \geq 3$ we suggest Algorithm 1 to compute an ordering of the simplices and vertices of a facet-adjacent triangulation, satisfying (O1) and (O2).

THEOREM 3.1. *Algorithm 1 determines in $\mathcal{O}(n^2 + nd^2)$ an ordering of the simplices and vertices satisfying (O1) and (O2).*

Proof. To proof the correctness of Algorithm 1, we first show that before and after each iteration of the while loop, the elements of the triangulation, formed by the simplices in \mathcal{S}' , are ordered according to (O1) and (O2) such that the ordering induces a hamiltonian cycle on the adjacency graph of \mathcal{S} . Second, we show that the cardinality of \mathcal{S}' is increased by one during each iteration of the loop.

In the first step of Algorithm 1, we have to choose two d -simplices $S_1, S_2 \in \mathcal{S}$ with $|\mathcal{V}(S_1) \cap \mathcal{V}(S_2)| = d$. Since the simplices in \mathcal{S} form a facet-adjacent triangulation, this is always possible and we can label the vertices in $\mathcal{V}(S_1) \cup \mathcal{V}(S_2)$ such that $\bar{x}_d^{S_1} = \bar{x}_0^{S_2}$ and $\bar{x}_0^{S_1} = \bar{x}_d^{S_2}$ holds. In the next step S_1 and S_2 are added to \mathcal{S}' , which is obviously ordered according to (O1) and (O2).

Since \mathcal{S} forms a facet-adjacent triangulation, there must be at least one simplex $S_i \in \mathcal{S}'$ which has a common facet (and thus d common vertices) with some simplex $S \in \mathcal{S} \setminus \mathcal{S}'$ at the beginning of each iteration of the loop. At the end of the while loop S is added to \mathcal{S}' . Thus, Algorithm 1 is finite.

Data: A set \mathcal{S} of d -simplices ($d \geq 3$), forming a facet-adjacent triangulation

Result: An ordering of the simplices $S \in \mathcal{S}$ and vertices $v \in \mathcal{V}(S)$ for all $S \in \mathcal{S}$, satisfying (O1) and (O2)

Choose two d -simplices $S_1, S_2 \in \mathcal{S}$ with $|\mathcal{V}(S_1) \cap \mathcal{V}(S_2)| = d$;

Label the vertices in $\mathcal{V}(S_1) \cup \mathcal{V}(S_2)$ such that $\bar{x}_d^{S_1} = \bar{x}_0^{S_2}$ and $\bar{x}_0^{S_1} = \bar{x}_d^{S_2}$ holds;

$\mathcal{S}' = (S_1, S_2)$;

while $|\mathcal{S}'| \neq n$ **do**

 Choose simplices $S \in \mathcal{S} \setminus \mathcal{S}'$ and $S_i \in \mathcal{S}'$ with $|\mathcal{V}(S) \cap \mathcal{V}(S_i)| = d$;

 Choose some vertex $w \in (\mathcal{V}(S) \cap \mathcal{V}(S_i)) \setminus \{\bar{x}_0^{S_i}, \bar{x}_d^{S_i}\}$;

if $\bar{x}_0^{S_i} \in \mathcal{V}(S)$ **then**

 Set $\bar{x}_0^S = \bar{x}_0^{S_i}$ and $\bar{x}_d^S = w$;

 Change the vertex labeling of S_i by setting $\bar{x}_0^{S_i} = w$;

 Set $\mathcal{S}' = (\dots, S_{i-1}, S, S_i, S_{i+1}, \dots)$;

else

 Set $\bar{x}_0^S = w$ and $\bar{x}_d^S = \bar{x}_d^{S_i}$;

 Change the vertex labeling of S_i by setting $\bar{x}_d^{S_i} = w$;

 Set $\mathcal{S}' = (\dots, S_{i-1}, S_i, S, S_{i+1}, \dots)$;

end

end

return \mathcal{S}'

Algorithm 1: Ordering d -simplices.

To conclude that the simplices in \mathcal{S}' are ordered appropriately at the end of each iteration, we have to see that there always exists some vertex $w \in (\mathcal{V}(S) \cap \mathcal{V}(S_i)) \setminus \{\bar{x}_0^{S_i}, \bar{x}_d^{S_i}\}$, because S and S_i have $d \geq 3$ common vertices. Further, there is only one vertex in $\mathcal{V}(S) \setminus \mathcal{V}(S_i)$ and $\mathcal{V}(S_i) \setminus \mathcal{V}(S)$, respectively. Therefore we either have $\bar{x}_0^{S_i} \in \mathcal{V}(S) \cap \mathcal{V}(S_i)$ or $\bar{x}_d^{S_i} \in \mathcal{V}(S) \cap \mathcal{V}(S_i)$ (or both).

In the first case, we insert S between S_i and its predecessor in \mathcal{S}' by setting $\bar{x}_0^S = \bar{x}_0^{S_i}$ and $\bar{x}_d^S = \bar{x}_0^{S_i} = w$ and leaving all other vertices untouched. In the second case we insert S between S_i and its successor analogously. Thus, in both cases \mathcal{S}' is ordered according to (O1) and (O2) at the end of each iteration.

Therefore, Algorithm 1 terminates after a finite number of steps, with an ordering of the simplices and vertices, satisfying properties (O1) and (O2).

Concerning the running time, it is easy to see that the while loop is executed n times and choosing a pair of simplices $S \in \mathcal{S} \setminus \mathcal{S}'$ and $S_i \in \mathcal{S}'$ with common facet, can be done in $\mathcal{O}(n)$ steps. Updating the vertex labels only depends on the dimension d and can be done in time $\mathcal{O}(d^2)$. All other operations can be done in constant time if appropriate data structures are used. Thus, we can conclude the running time of Algorithm 1 to be $\mathcal{O}(n^2 + nd^2)$. \square

We like to remark that a consequence of Theorem 3.1 is that every graph, which is an adjacency graph of a facet-adjacent triangulation with d -simplices ($d \geq 3$) has a hamiltonian cycle.

The third MIP model for piecewise linear multivariate functions we present is the *logarithmic disaggregated convex combination model* from Vielma and Nemhauser [23]. Their approach is based on the disaggregated convex combination model but uses just a logarithmic number of auxiliary binary variables to enforce a single simplex to be active. The idea for reducing the amount of binary variables results from the fact that $\lceil \log_2(n) \rceil$ binary variables are sufficient to encode n different states, each choosing one simplex out of n . Therefore an arbitrary injective function $c : \mathcal{S} \rightarrow \{0, 1\}^{\lceil \log_2 n \rceil}$ can be used.

$$x = \sum_{i=1}^n \sum_{j=0}^d \lambda_j^{S_i} \bar{x}_j^{S_i}, \quad \sum_{i=1}^n \sum_{j=0}^d \lambda_j^{S_i} = 1, \quad \lambda \geq 0, \quad (3.12)$$

$$y = \sum_{i=1}^n \sum_{j=0}^d \lambda_j^{S_i} \bar{y}_j^{S_i}, \quad (3.13)$$

$$\sum_{i=1}^n \sum_{j=0}^d c(S_i)_l \lambda_j^{S_i} \leq z_l \quad \text{for } l = 1, \dots, \lceil \log_2 n \rceil, \quad (3.14)$$

$$\sum_{i=1}^n \sum_{j=0}^d (1 - c(S_i)_l) \lambda_j^{S_i} \leq 1 - z_l \quad \text{for } l = 1, \dots, \lceil \log_2 n \rceil, \quad (3.15)$$

$$z_l \in \{0, 1\} \quad \text{for } l = 1, \dots, \lceil \log_2 n \rceil. \quad (3.16)$$

Constraints (3.12) and (3.13) present the point (x, y) as convex combination of given simplices and remain unchanged against the disaggregated

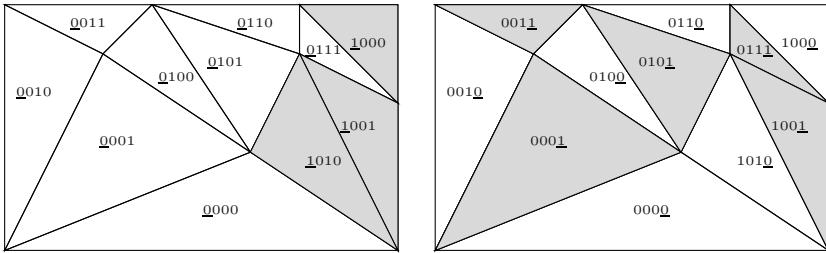


FIG. 8. Example for a binary encoding and the branching induced by the leftmost and rightmost bit using the logarithmic disaggregated convex combination model.

convex combination model. This model is another locally ideal formulation. An illustration of an example binary encoding and a branching on the leftmost and rightmost bit is depicted in Fig. 8. In each case white triangles correspond to a down branch and the up branch is shown in gray.

For a generalization to higher dimensions of the aggregated *logarithmic convex combination model* from the univariate case we refer to Vielma and Nemhauser [23]. We skip this model because it is not as flexible as the others in the sense that it requires the set of simplices \mathcal{S} to be topologically equivalent to the J_1 triangulation, which is also known as “Union Jack”-triangulation, and thus cannot be applied to arbitrary triangulations like the other formulations.

4. Controlling the approximation error. In the preceding sections we have presented a couple of methods to model piecewise linear functions in terms of mixed integer linear constraints. Although we have seen that there are applications, in which piecewise linear functions arise naturally, it is also possible to apply the above mentioned techniques to piecewise linear approximations of nonlinear expressions. By this means, we are able to formulate MIP approximations of (mixed integer) nonlinear programs.

In order to make a point about the quality of such an approximation, we must be able to measure the linearization errors. Doing this a-posteriori is quite simple: We solve some MIP approximation to optimality and evaluate the nonlinear expressions at the point where the optimum is attained. Unfortunately, this is not enough, whenever we want the solution to satisfy some predefined error bounds. In this case we must be able to estimate the linearization errors a-priori in order to avoid solving MIPs of increasing complexity (due to finer linearization grids) again and again.

In this section, we will introduce a general methodology to construct a-priori error estimators for piecewise linear approximations.

As we have seen the complexity of the resulting MIPs strongly depends on the number of linear pieces of the approximations and if we overestimate the linearization errors, we will need more linear pieces as necessary to

ensure that the error bounds are satisfied. Hence, we are interested in as strong as possible error estimators.

As starting point for our considerations, we assume the following situation: Let $\mathcal{D} \subseteq \mathbb{R}^d$ and $f : \mathcal{D} \rightarrow \mathbb{R}$ be some continuous function. Further, let $\phi : \mathcal{P} \rightarrow \mathbb{R}$ be a piecewise linear approximation of f over some convex polytope $\mathcal{P} \subseteq \mathcal{D}$. We assume ϕ to interpolate f on the vertices of some triangulation of \mathcal{P} . Thus, for a triangulation with simplex set \mathcal{S} , we can define affine functions $\phi_i : x \rightarrow a_i^T x + b_i$ for each simplex $S_i \in \mathcal{S}$ with $\phi_i(x) = f(x)$ for every vertex x of S_i such that we can write $\phi(x) = \phi_i(x)$ for $x \in S_i$.

If we are able to control the linearization error within a simplex, we are obviously able to control the overall linearization error, since we can simply add a point of maximum error to the vertex set of our triangulation and retriangulate the affected region locally. Repeating this process for all not yet checked simplices, leads to a piecewise linearization, which satisfies the error bound everywhere.

For this reason, we restrict our further considerations to the situation, where $\phi(x) = a^T x + b$ is the linear interpolation of f over a simplex S with vertices $\bar{x}_0, \dots, \bar{x}_d$, i.e., $\phi(\bar{x}_i) = f(\bar{x}_i)$ for $i = 1, \dots, d$. We define the maximum linearization error in terms of the maximum under- and overestimation of a function (cf. Fig. 9):

DEFINITION 4.1. *We call $\epsilon_u(f, S) := \max_{x \in S} f(x) - \phi(x)$ the maximum underestimation, $\epsilon_o(f, S) := \max_{x \in S} \phi(x) - f(x)$ the maximum overestimation and $\epsilon(f, S) = \max\{\epsilon_u(f, S), \epsilon_o(f, S)\}$ the maximum linearization error of f by ϕ over S .*

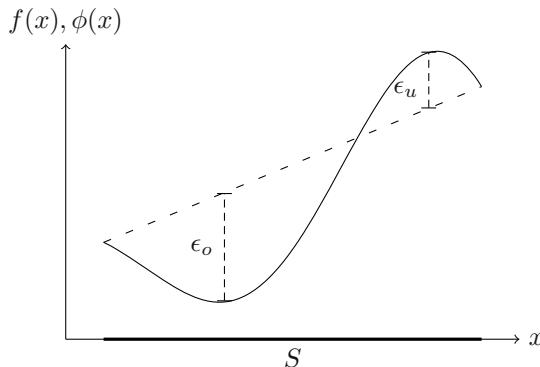


FIG. 9. *The maximum underestimation ϵ_u and the maximum overestimation ϵ_o of f by ϕ over S .*

Since the maximum overestimation $\epsilon_o(f, S)$ of f over S with respect to ϕ is equal to the maximum underestimation $\epsilon_u(-f, S)$ of $-f$ over S with respect to $-\phi$, we restrict ourselves to only consider the maximum

overestimation for the remainder of this section. In cases, where f is convex (or concave) over S , calculating the maximum overestimation can be done efficiently as a consequence of the following proposition:

PROPOSITION 4.1. *If f is convex over S , then $\epsilon_o(f, S)$ can be obtained by solving a convex optimization problem. If f is concave over S , then $\epsilon_o(f, S) = 0$ holds.*

Proof. If f is convex, we have $\epsilon_o(f, S) = \max_{x \in S} \phi(x) - f(x)$. Since f is convex, $-f$ is concave and thus, $\phi(x) - f(x)$ is concave as the sum of concave functions. Therefore, $f(x) - \phi(x)$ is convex and we can calculate $\epsilon_o(f, S) = -\min_{x \in S} f(x) - \phi(x)$ by solving a convex minimization problem over a simplicial domain.

Next, we consider the case, where f is concave. Since ϕ is an affine function, for every $x \in S$, the point $(x, \phi(x))$ is a convex combination of the vertices $\bar{x}_0, \dots, \bar{x}_d$ of S and the associated values of ϕ , i.e.,

$$x = \sum_{i=0}^d \lambda_i \bar{x}_i, \quad \phi(x) = \sum_{i=0}^d \lambda_i \phi(\bar{x}_i), \quad \sum_{i=0}^d \lambda_i = 1, \quad \lambda_i \geq 0, \quad i = 0, \dots, d.$$

Since f is concave, we can apply Jensen's inequality [12] to get

$$f(x) = f\left(\sum_{i=0}^d \lambda_i \bar{x}_i\right) \geq \sum_{i=0}^d \lambda_i f(\bar{x}_i).$$

Since ϕ interpolates f in the vertices of S , we get

$$f(x) \geq \sum_{i=0}^d \lambda_i \phi(\bar{x}_i) = \phi(x),$$

which shows $\epsilon_o(f, S) = 0$. □

Thus, we only have to maximize a concave function (i.e., minimize a convex function) over a simplicial domain, in order to obtain the maximum overestimation, whenever we know that the approximated function is either convex or concave over S . This can be done efficiently. Unfortunately, the situation gets more involved, if f is indefinite, or if we simply do not know about the definiteness of f over S . In this case, we can only estimate the maximum overestimation in general. To show, how to do this, we need the following definitions:

DEFINITION 4.2.

(i) *A function*

$$\mu \in \mathcal{U}(f, S) := \{\xi : S \rightarrow \mathbb{R} : \xi \text{ convex}, \xi(x) \leq f(x) \forall x \in S\}$$

is called a convex underestimator of f over S .

(ii) The function $\text{vex}_S[f] : S \rightarrow \mathbb{R}$, defined as

$$\text{vex}_S[f](x) := \sup\{\mu(x) : \mu \in \mathcal{U}(f, S)\}$$

is called the convex envelope of f over S .

Clearly, the convex envelope is the tightest convex underestimator of f over S , since the pointwise supremum of convex functions is convex [26].

To derive upper bounds for the maximum overestimation of f over S , we can use convex underestimating functions as shown in the following proposition.

PROPOSITION 4.2. *Let μ be a convex underestimating function of f over S , then $\epsilon_o(f, S) \leq \epsilon_o(\mu, S)$ holds.*

Proof. Since $\mu(x) \leq f(x) \quad \forall x \in S$, we get

$$\epsilon_o(f, S) = \max_{x \in S} \phi(x) - f(x) \leq \max_{x \in S} \phi(x) - \mu(x) = \epsilon_o(\mu, S). \quad \square$$

There are many cases, in which we can derive a convex underestimator for an indefinite function. For example, in the case of a twice continuously differentiable function, we can use α -underestimators as introduced in [20] or a further developed class of convex underestimators, based on piecewise α -underestimators as described in [9, 10]. As we mentioned above, the complexity of the approximating mixed integer linear programs highly depend on the quality of the error estimators. Naturally, the best possible error estimators can be constructed by using the convex envelopes of the approximated function. From Theorem 4.1 below, we will see that using the envelopes, we can actually calculate the exact value of the maximum linearization error within a simplex. Before we formulate the theorem we give two lemmas summarizing some basic properties of convex envelopes:

LEMMA 4.1. *Let $\mathcal{C} \subset \mathbb{R}^d$ be a subset of the d -dimensional real space, $\xi, \eta : \mathcal{C} \rightarrow \mathbb{R}$ real valued functions and η affine, then for all $x \in \text{conv}(\mathcal{C})$ $\text{vexc}_{\mathcal{C}}[\xi](x) + \text{vexc}_{\mathcal{C}}[\eta](x) = \text{vexc}_{\mathcal{C}}[\xi + \eta](x)$.*

LEMMA 4.2. *Let \mathcal{C} be a compact subset of \mathbb{R}^d and ξ a lower semi-continuous real-valued function on \mathcal{C} . Further, let \mathcal{M} be the set of globally minimum points of ξ over \mathcal{C} and \mathcal{N} the set of globally minimum points of $\text{vexc}_{\mathcal{C}}[\xi]$. Then*

$$\min_{x \in \mathcal{C}} \xi(x) = \min_{x \in \text{conv}(\mathcal{C})} \text{vexc}_{\mathcal{C}}[\xi](x) \quad \text{and} \quad \mathcal{N} = \text{conv}(\mathcal{M}).$$

Here, the convex hull of a set S is denoted by $\text{conv}(S)$. A proof of Lemma 4.1 and Lemma 4.2 can be found in [28]. Now we can formulate Theorem 4.1, which constitutes the main result of this section. We will see that once we have the convex envelope of f , we are able to calculate the maximum overestimation by solving a convex optimization problem.

Moreover, we can identify a point, where the maximum error is attained efficiently.

THEOREM 4.1. *Let $\mathcal{M}_o := \{x \in S : \phi(x) - f(x) = \epsilon_o(f, S)\}$ be the set of global maximizers for the overestimation of f by ϕ over S and let $\mathcal{N}_o := \{x \in S : \phi(x) - \text{vex}_S[f](x) = \epsilon_o(\text{vex}_S[f], S)\}$ be the set of global maximizers for the overestimation of the convex envelope of f by ϕ over S . Then we get $\epsilon_o(f, S) = \epsilon_o(\text{vex}_S[f], S)$ and $\mathcal{N}_o = \text{conv}(\mathcal{M}_o)$.*

Proof. We get

$$\begin{aligned}\epsilon_o(\text{vex}_S[f], S) &= \max_{x \in S} \phi(x) - \text{vex}_S[f](x) \\ &= -\min_{x \in S} \text{vex}_S[f](x) - \phi(x) \\ &= -\min_{x \in S} \text{vex}_S[f](x) + \text{vex}_S[-\phi](x),\end{aligned}$$

since $-\phi$ is affine. Next, we can apply Lemma 4.1 to conclude

$$\epsilon_o(\text{vex}_S[f], S) = -\min_{x \in S} \text{vex}_S[f - \phi](x).$$

By assumption, S is convex and we can use Lemma 4.2 to get

$$\begin{aligned}\epsilon_o(\text{vex}_S[f], S) &= -\min_{x \in S} (f - \phi)(x) \\ &= -\min_{x \in S} f(x) - \phi(x) \\ &= \max_{x \in S} \phi(x) - f(x) \\ &= \epsilon_o(f, S).\end{aligned}$$

The identity $\mathcal{N}_o = \text{conv}(\mathcal{M}_o)$, again follows from Lemma 4.2, which completes the proof. \square

In order to calculate a point where the maximum overestimation of f by ϕ over S is attained, it suffices to solve the convex optimization problems

$$\min_{x \in \mathcal{N}_o} \{x_i : x_j = x_j^j, j < i\} \quad (4.1)$$

iteratively for $i = 1, \dots, d$. Here x^j is the solution of Problem (4.1) for $i = j$. The point x^j lies on all supporting hyperplanes $\mathcal{H}^k = \{x \in \mathbb{R}^d : x_k = x_k^k\}$ for $k \leq j$ of \mathcal{N}_o . Since the normal vectors of $\mathcal{H}^1, \dots, \mathcal{H}^j$ are linearly independent, the dimension of $\bigcap_{k \leq j} (\mathcal{H}^k) \cap \mathcal{N}_o$ is at most $d - j$. Thus, $x^d = \bigcap_{i=1}^d (\mathcal{H}^i) \cap \mathcal{N}_o$ must be an extreme point of \mathcal{N}_o and we get $x^d \in \mathcal{M}_o$. Therefore, x^d is a global minimizer of $\phi(x) - f(x)$. We remark that it is not necessary to solve d instances of Problem (4.1) in general, since if an optimal solution x^j of (4.1) for $j < d$ satisfies $\phi(x^j) - f(x^j) = \epsilon_o(\text{vex}_S[f], S)$, the point x^j already is an extreme point of the convex hull of \mathcal{M}_o .

Unfortunately it is hard to determine the envelopes of indefinite functions in general, but in recent times, there was much progress in this field,

e.g., in [11] it was shown that the envelopes of indefinite $(d - 1)$ -convex functions can be calculated efficiently. We have to point out that although Theorem 4.1 is valid for any dimension, it is impractical to apply the techniques described in this chapter to nonlinear expressions, which depend on a large number of variables, because even the number τ_d of simplices, necessary to triangulate the d -cube is growing rapidly with d . From [27], we know that $\tau_d \geq \frac{6^{\frac{d}{2}} \cdot d!}{2 \cdot (d+1)^{\frac{d+1}{2}}}$ is a valid inequality. Even though, we can only deal with piecewise linear approximations of nonlinear expressions depending just on a few variables efficiently, the vast majority of nonlinearities, occurring in practical applications are of this type or they are separable such that they can be decomposed into uni-, bi- or trivariate functions.

In the following section, we will show, how to use the error estimators developed so far and the modeling techniques from the preceding sections to construct a mixed integer linear programming relaxation for any given mixed integer nonlinear program.

5. MIP relaxations of mixed integer nonlinear programs. In this section we will show how to construct a mixed integer linear programming relaxation for a mixed integer nonlinear program. To this end we will slightly modify the techniques introduced in Section 2 and 3 by using the error estimators developed in Section 4. In the remainder of this chapter we deal with a mixed integer nonlinear program of the form

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & g_i(x) = b_i \quad \text{for } i = 1, \dots, k_1, \\ & h_i(x) \geq b_i \quad \text{for } i = 1, \dots, k_2, \\ & l \leq x \leq u, \\ & x \in \mathbb{R}^{d-p} \times \mathbb{Z}^p, \end{aligned} \tag{5.1}$$

where $g_i : \mathbb{R}^d \rightarrow \mathbb{R}$ for $i = 1, \dots, k_1$ and $h_i : \mathbb{R}^d \rightarrow \mathbb{R}$ for $i = 1, \dots, k_2$ are real valued functions and $l, u \in \mathbb{R}^d$ are the vectors of lower and upper bounds on the variables.

Our goal is to construct piecewise polyhedral envelopes as shown in Fig. 10 for each g_i and h_i using modified versions of the modeling techniques for piecewise linear functions introduced in Section 2 and 3. For $f = g_i$ and $i \in \{1, \dots, k_1\}$ the idea is as simple as introducing an additional real valued variable e , which is added to the approximated function value and which is bounded from below and above by $-\epsilon_o(f, S)$ and $\epsilon_u(f, S)$ (cf. Section 4), respectively. These variable bounds depend on the simplex S and we will explain how to model these bounds appropriately. To this end we consider some (nonlinear) expression $f = g_i$ occurring on the left-hand side in one of the equality constraints of Problem (5.1), together with its piecewise linear approximation ϕ (cf. Section 4).

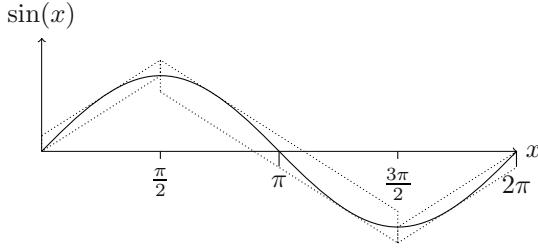


FIG. 10. Piecewise polyhedral envelopes of $\sin(x)$ on $[0, 2\pi]$ with breakpoints $0, \frac{\pi}{2}, \frac{3\pi}{2}, 2\pi$.

In case of the convex combination, logarithmic or SOS model, the approximated function value is given by Equation (3.3), which we modify as follows:

$$y = \sum_{j=1}^m \lambda_j \bar{y}_j + e. \quad (5.2)$$

In case of the disaggregated logarithmic model, we modify equation (3.13) to

$$y = \sum_{i=1}^n \sum_{j=0}^d \lambda_j^{S_i} \bar{y}_j^{S_i} + e. \quad (5.3)$$

Additionally we introduce the following two inequalities, to model the bounds of e in case of the convex combination model:

$$-\sum_{i=1}^n \epsilon_o(f, S_i) z_i \leq e \leq \sum_{i=1}^n \epsilon_u(f, S_i) z_i. \quad (5.4)$$

In a logarithmic or an SOS model, we can only incorporate the proper bounds during branching. For the disaggregated logarithmic model we guarantee the bounds on e by

$$-\sum_{i=1}^n \left(\epsilon_o(f, S_i) \sum_{j=0}^d \lambda_j^{S_i} \right) \leq e \leq \sum_{i=1}^n \left(\epsilon_u(f, S_i) \sum_{j=0}^d \lambda_j^{S_i} \right), \quad (5.5)$$

where $\epsilon_u(f, S_i)$ and $\epsilon_o(f, S_i)$ are computed as described in Section 4. In every feasible solution, the λ -variables sum up to 1 and satisfy the SOS condition (i.e., all positive λ -variables belong to vertices of the same simplex). The boxes, depicted in Fig. 10 exactly describe the feasible region of the MIP resulting from the modified convex combination, SOS or (disaggregated) logarithmic model together with (5.4) and (5.5), respectively.

If $f = h_i$ for some $i \in \{1, \dots, k_2\}$ we can omit the left inequality in (5.4) and (5.5), because we must not bound e from above.

In case of the incremental method we substitute equation (3.7) by

$$y = \bar{y}_0^{S_i} + \sum_{i=1}^n \sum_{j=1}^d \left(\bar{y}_j^{S_i} - \bar{y}_0^{S_i} \right) \delta_j + e \quad (5.6)$$

and add the inequalities

$$\epsilon_u(f, S_1) + \sum_{i=1}^{n-1} z_i (\epsilon_u(f, S_{i+1}) - \epsilon_u(f, S_i)) \geq e, \quad (5.7)$$

$$-\epsilon_o(f, S_1) - \sum_{i=1}^{n-1} z_i (\epsilon_o(f, S_{i+1}) - \epsilon_o(f, S_i)) \leq e. \quad (5.8)$$

The feasible region of the MIP described by the modified incremental model together with the Constraints (5.7) and (5.8) is again the union of the boxes, depicted in Fig. 10. In contrast to the modified disaggregated logarithmic method, we can only model the piecewise polyhedral envelopes appropriately, if we add inequalities containing binary variables. To clarify the correctness of (5.7) and (5.8) remember that in every feasible solution of the described MIP there is some index j with $z_i = 1$ for all $i < j$ and $z_i = 0$ for all $i \geq j$. This means that all terms $\epsilon_u(f, S_i)$ on the left-hand side of (5.7) and all terms $\epsilon_o(f, S_i)$ on the left-hand side of (5.8) with $i \neq j$ either cancel out or are multiplied by 0. Therefore, we get $-\epsilon_o(f, S_j) \leq e \leq \epsilon_u(f, S_j)$ as desired.

As we have seen, it suffices to introduce $k_1 + k_2$ additional real valued variables more to model a mixed integer piecewise polyhedral relaxation instead of a mixed integer piecewise linear approximation of (5.1). This marginal overhead makes every feasible point of (5.1) also feasible for its MIP relaxation. Thus, we can use any NLP solver to produce feasible solutions for the MIP, once the integer variables x_{d-1+p}, \dots, x_d are fixed. If we only accept solutions, feasible for the MINLP, as incumbent solutions of the MIP relaxation, it is straightforward to implement an algorithm to solve (5.1), which can proof optimality and prune infeasible or suboptimal parts of the branch and bound tree by using the whole bunch of well evolved techniques integrated in modern solvers for mixed integer linear problems. Even if we think about such an MIP relaxation without having any global solver for the underlying MINLP in mind, there are some obvious advantages compared to a (piecewise linear) approximation. First, any solution of an MIP relaxation (and even any solution to a corresponding LP relaxation) yields a valid lower bound for the MINLP. On the other hand, if we consider just an approximation, no such statement can be made without making further assumptions concerning the convexity of the involved

nonlinearities. Second, even piecewise linear MIP approximations of high accuracy are likely to be infeasible, although the underlying MINLP has a nonempty set of feasible points. This becomes apparent e.g., for MINLPs having constraints of type $f(x) = g(x)$, where f and g are nonlinear functions, whose graphs intersect in a finite number of feasible points. For an MIP relaxation all these points are feasible regardless of the chosen accuracy. On the other hand, even a low accuracy approximation might capture such points of intersection rather exact and thus reflects the properties of the MINLP model quite well, while a relaxation might smooth the feasible set in such a way that the resulting solutions violate some characteristic properties of the underlying MINLP.

In [18] a technique similar to the approach presented above has been introduced. There, the piecewise polyhedral envelopes are constructed using special ordered sets but instead of branching to fulfill the SOS condition, they branch on the original problem variables and introduce new breakpoints in each new node of the branch and bound tree. The second difference is that they decompose the involved nonlinear expressions into simpler expressions and derive explicit formulas for the maximum linearization error for a set of nonlinear expressions occurring within the problem of power system analysis. Therefore our approach to calculate the linearization errors is somewhat more general. On the other hand we are not yet able to refine the polyhedral envelopes dynamically, but we believe that the combinatorics of piecewise linear MIP models can be further exploited by appropriate cutting planes and separation algorithms.

6. Computational results. In this section we report on some computational results for problems including the introduced piecewise linearization techniques. These problems are based upon the real-world applications of water supply network optimization and transient technical optimization of gas networks that consider the problem of time-dependent optimization in distribution networks. These networks consist of pipes to transport water or gas from suppliers to consumers. Due to friction within pipes pressure gets lost. This loss can be compensated by pumps in water supply networks or compressor stations in the case of gas transportation. Running pumps and compressors consume power depending on their current pressure increase and flow. This power is either generated by electricity or by the combustion of fuel gas. The aim of our optimization problems is to minimize the overall electricity and fuel gas consumption, respectively.

After applying a time and space discretization this problem reduces to a mixed integer nonlinear program. On the one hand nonlinearities are used to represent the physical behavior of water or gas within the network components and on the other hand binary variables are used to describe discrete switching processes of valves, compressors and pumps or to enforce minimum running times of the devices.

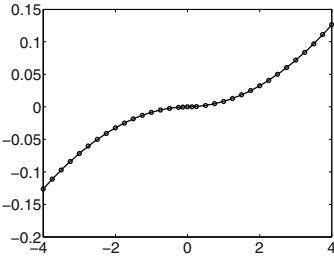


FIG. 11. Piecewise linear approximation of the solution set for equation (6.1).

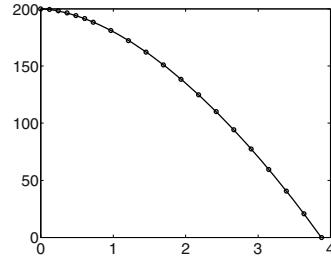


FIG. 12. Piecewise linear approximation of the solution set for equation (6.3).

We formulate an MIP to solve the problem of minimal power consumption of gas and water networks, respectively. Nonlinearities are approximated by piecewise linearizations and included in our models by applying the methods from Section 2 and Section 3. Afterward CPLEX as black box MIP solver is used to solve the resulting mixed binary linear models. We refer to [19] for further details on the problem of transient technical optimization.

The arising nonlinearities of our water network optimization problems are the pressure loss due to friction within pipes, which can be described by equations of the form

$$y_1 = \lambda x_1 |x_1|. \quad (6.1)$$

The factor λ is implicitly defined through an equation

$$\frac{1}{\sqrt{\lambda}} = -a \log_{10} \left(b + \frac{c}{x_1 \sqrt{\lambda}} \right), \quad (6.2)$$

with constants a, b and c . Additionally, a pump's pressure increase is either defined by a smooth function of the form

$$y_2 = dx_2^e, \quad (6.3)$$

with constant values d and e or directly by a piecewise linear interpolation of a given point set. All the above nonlinearities are univariate and we approximate them by an interpolation as explained in Section 4. For instance, for a water network optimization problem instance with eight pipes, four pumps and 24 time steps we get 192 nonlinearities of the form (6.1) and 96 of the type (6.3) in our chosen example. On average each of them is approximated by about 30 line segments (cf. Fig. 11 and Fig. 12). The accuracy is guaranteed through a relative approximation error of at most 10^{-3} .

TABLE 1
Demonstration of univariate models for a water supply network example.

	cc	inc	sos	log	dlog
No. of vars	6522	6621	3871	4697	7526
No. of ints	2851	3001	200	776	776
No. of aux	2751	2901	150	676	676
No. of cons	3920	9419	569	2145	2149
No. of nodes	136706	16004	182161	85383	128320
running time	1200	106	413	1462	* (∞)

For all numerical results we use ILOG CPLEX 11.2 with default parameters and we do not interfere with the solution process. All computations are done on one core of an Intel Core2Duo 2GHz machine with 2GB RAM running on a Linux operating system. The maximum running time is restricted to at most one hour. We omit the time to calculate the piecewise linear interpolations, because in our examples it is insignificant compared to CPLEX' running time. In Table 1 we compare all mixed integer formulations from Section 2 on a water network optimization problem. The table shows the entire number of variables, integer variables, auxiliary binary variables and constraints of the resulting model as well as the number of branch-and-bound nodes and solution times in seconds CPLEX needed to prove optimality. An asterisk indicates that the time limit was reached. In this case the solver's gap at that time is printed in parentheses. The model abbreviations stand for convex combination (cc), incremental (inc), special ordered sets (sos), logarithmic convex combination (log) and logarithmic disaggregated convex combination (dlog) model. We see that even if sos, log and dlog yield smaller models it is not always a good idea to use just these models. In our example inc is by far the best choice, a result that agrees with further examinations of water network supply instances. In addition to the results computed by piecewise linear approximations we also compare them to the piecewise linear relaxations introduced in Section 5. For our example the running times of both approaches do not differ very much. In general one could say it is slightly more difficult to find a feasible solution using the approximation techniques, whereas the optimality proof is typically somewhat faster. More interesting, however, is an investigation of the optimal solutions found by both approaches (see Table 3). The relative difference of the objective values obtained by piecewise linear approximation and piecewise linear relaxation is 10^{-3} , which is not greater than our error tolerance for the nonlinearities. Of course we cannot ensure this to hold for arbitrary instances or models, but as our examples and experiences show it is commonly true for our problems. The exact objective value of this global optimum is somewhere in between of those achieved by interpolation and relaxation.

In gas network optimization problems, higher dimensional nonlinearities arise, too. The friction within pipes is expressed by a bivariate nonlinear equation

$$y_1 = c \lambda \frac{x_1^2}{x_2}, \quad (6.4)$$

with constant factor c and λ defined again by equation (6.2). The 32 arising equations of this form are replaced by piecewise linearizations with approximately 14 triangles to achieve a maximum relative error of at most 1%. Furthermore, trivariate nonlinear terms arise to describe the compressor's power consumption through an equation of the form

$$y_2 = a x_1 \left(\left(\frac{x_2}{x_3} \right)^b - c \right), \quad (6.5)$$

where a, b and c are constants. Each of the arising ten equations of the above type are piecewise linearized with approximately 700 tetrahedra. In [Table 2](#) computational results for the three different formulations from Section 3, namely convex combination (cc), incremental (inc) and logarithmic disaggregated convex combination (dlog) model are listed for a gas network optimization problem. We see for our representative example that dlog plays out its advantage of the drastically reduced number of binary variables and constraints. Again, inc models seem to have the best structure, but the LPs are far too large compared to the others. A comparision of the results obtained by piecewise linear relaxations shows almost identical running times to those achieved by interpolation techniques. As shown in [Table 3](#) the relative difference between both objective values is approximately 2.7%, which is in the same order of magnitude as our specified maximum approximation error of the nonlinearities. Likewise, we see that the solution found by piecewise linearization techniques lies within 1.3% and 1.4% of the exact optimum. In addition we like to mention that all crucial decisions, i.e. problem-specific integer variables that are not introduced to model piecewise linearizations, have an identical assignment in all presented cases.

7. Future directions. We have seen that using piecewise linear approximations can be an attractive way of tackling mixed integer nonlinear programs. We get a globally optimal solution within a-priori determined tolerances and are able to use the well-developed software tools of mixed integer linear programming. So far, the best case where these techniques can be used is when there are only few different nonlinear functions each of which depends only on a few variables. Then the combinatorics of the problem dominate the complexity of the approximation and we can expect a huge gain in being able to use MIP techniques.

This also opens up a variety of different directions for further developing these methods. One important topic is to fuse the techniques shown

TABLE 2
Demonstration of multivariate models for a gas network example.

	cc	inc	dlog
No. of vars	10302	30885	29397
No. of ints.	7072	7682	362
No. of aux.	7036	7646	326
No. of cons	3114	23592	747
No. of nodes	620187	18787	75003
running time	* (0.81%)	3204	1386

TABLE 3
Comparison between optimal objective values obtained by piecewise approximation, piecewise relaxation and exact optimum.

	Approximation	Relaxation	Exact
Water network example	73.4473	73.3673	73.4211
Gas network example	2.7661	2.6903	2.7268

in this chapter with classical nonlinear programming techniques. Can we find a piecewise linear approximation that is just fine enough to get the optimal setting of the integral variables? Then we could fix the integral variables and use a standard nonlinear programming tool to compute the optimal value of the continuous variables. As the schemes we have presented are quite general and do not require much of the functions to be approximated, it will be important in which cases one can do better. In the case of a second-order cone constraint Ben-Tal and Nemirovski [5] have given an ingenious construction for an LP model of a piecewise linear approximation that does not need any extra binary variables. What about other functions, can we give similar constructions for those? Modeling support will also be an important practical topic: How can we relieve the modeler of the burden to generate a good approximation of the nonlinear function and automate this process?

REFERENCES

- [1] A. BALAKRISHNAN AND S.C. GRAVES, *A composite algorithm for a concave-cost network flow problem*, Networks, **19** (1989), pp. 175–202.
- [2] J.J. BARTHOLDI III AND P. GOLDSMAN, *The vertex-adjacency dual of a triangulated irregular network has a hamiltonian cycle*, Operations Research Letters, **32** (2004), pp. 304–308.
- [3] E.M.L. BEALE AND J.J.H. FORREST, *Global optimization using special ordered sets*, Math. Programming, **10** (1976), pp. 52–69.
- [4] E.M.L. BEALE AND J.A. TOMLIN, *Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables*,

- in OR 69, J. Lawrence, ed., International Federation of Operational Research Societies, Travistock Publications, 1970, pp. 447–454.
- [5] A. BEN-TAL AND A. NEMIROVSKI, *On polyhedral approximations of the second-order cone*, Math. Oper. Res., **26** (2001), pp. 193–205.
 - [6] K.L. CROXTON, B. GENDRON, AND T.L. MAGNANTI, *Variable disaggregation in network flow problems with piecewise linear costs*, Oper. Res., **55** (2007), pp. 146–157.
 - [7] G.B. DANTZIG, *On the significance of solving linear programming problems with some integer variables*, Econometrica, **28** (1960), pp. 30–44.
 - [8] I.R. DE FARIAS, JR., M. ZHAO, AND H. ZHAO, *A special ordered set approach for optimizing a discontinuous separable piecewise linear function*, Oper. Res. Lett., **36** (2008), pp. 234–238.
 - [9] C.E. GOUNARIS AND C.A. FLOUDAS, *Tight convex underestimators for C^2 -continuous problems: I. multivariate functions*, Journal of Global Optimization, **42** (2008), pp. 69–89.
 - [10] ———, *Tight convex underestimators for C^2 -continuous problems: I. univariate functions*, Journal of Global Optimization, **42** (2008), pp. 51–67.
 - [11] M. JACH, D. MICHAELS, AND R. WEISMANTEL, *The convex envelope of $(n-1)$ convex functions*, Siam Journal on Optimization, **19** (3) (2008), pp. 1451–1466.
 - [12] J.L.W.V. JENSEN, *Sur les fonctions convexes et les inégalités entre les valeurs moyennes*, Acta Mathematica, **30** (1) (1906), pp. 175 – 193.
 - [13] A.B. KEHA, I.R. DE FARIAS, JR., AND G.L. NEMHAUSER, *Models for representing piecewise linear cost functions*, Oper. Res. Lett., **32** (2004), pp. 44–48.
 - [14] ———, *A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization*, Oper. Res., **54** (2006), pp. 847–858.
 - [15] T. KOCH, *Personal communication*, 2008.
 - [16] A. KRION, *Optimierungsmethoden zur Berechnung von Cross-Border-Flow beim Market-Coupling im europäischen Stromhandel*, Master's thesis, Discrete Optimization Group, Department of Mathematics, Technische Universität Darmstadt, Darmstadt, Germany, 2008.
 - [17] J. LEE AND D. WILSON, *Polyhedral methods for piecewise-linear functions. I. The lambda method*, Discrete Appl. Math., **108** (2001), pp. 269–285.
 - [18] S. LEYFFER, A. SARTENAER, AND E. WANUFELL, *Branch-and-refine for mixed-integer nonconvex global optimization*, Tech. Rep. ANL/MCS-P1547-0908, Argonne National Laboratory, Mathematics and Computer Science Division, 2008.
 - [19] D. MAHLKE, A. MARTIN, AND S. MORITZ, *A mixed integer approach for time-dependent gas network optimization*, Optimization Methods and Software, **25** (2010), pp. 625 – 644.
 - [20] C.D. MARANAS AND C.A. FLOUDAS, *Global minimum potential energy conformations of small molecules*, Journal of Global Optimization, **4** (1994), pp. 135–170.
 - [21] H.M. MARKOWITZ AND A.S. MANNE, *On the solution of discrete programming problems*, Econometrica, **25** (1957), pp. 84–110.
 - [22] R.R. MEYER, *Mixed integer minimization models for piecewise-linear functions of a single variable*, Discrete Mathematics, **16** (1976), pp. 163 – 171.
 - [23] G. NEMHAUSER AND J.P. VIELMA, *Modeling disjunctive constraints with a logarithmic number of binary variables and constraints*, in Integer Programming and Combinatorial Optimization, Vol. **5035** of Lecture Notes in Computer Science, 2008, pp. 199–213.
 - [24] M. PADBERG, *Approximating separable nonlinear functions via mixed zero-one programs*, Oper. Res. Lett., **27** (2000), pp. 1–5.
 - [25] M. PADBERG AND M.P. RIJAL, *Location, scheduling, design and integer programming*, Kluwer Academic Publishers, Boston, 1996.
 - [26] R.T. ROCKAFELLAR, *Convex Analysis*, Princeton University Press, 1970.

- [27] W.D. SMITH, *A lower bound for the simplicity of the n -cube via hyperbolic volumes*, European Journal of Combinatorics, **21** (2000), pp. 131–137.
- [28] F. TARDELLA, *On the existence of polyhedral convex envelopes*, in Frontiers in global optimization, C. Floudas and P. M. Pardalos, eds., Vol. **74** of Nonconvex Optimization and its Applications, Springer, 2004, pp. 563 – 573.
- [29] M.J. TODD, *Hamiltonian triangulations of \mathbb{R}^n* , in Functional Differential Equations and Approximation of Fixed Points, A. Dold and B. Eckmann, eds., Vol. **730/1979** of Lecture Notes in Mathematics, Springer, 1979, pp. 470 – 483.
- [30] J.P. VIELMA, S. AHMED, AND G. NEMHAUSER, *Mixed-Integer models for nonseparable Piecewise-Linear optimization: Unifying framework and extensions*, Operations Research, **58** (2009), pp. 303–315.
- [31] J.P. VIELMA, A.B. KEHA, AND G.L. NEMHAUSER, *Nonconvex, lower semicontinuous piecewise linear optimization*, Discrete Optim., **5** (2008), pp. 467–488.
- [32] D. WILSON, *Polyhedral methods for piecewise-linear functions*, Ph.D. thesis in Discrete Mathematics, University of Kentucky, 1998.

AN ALGORITHMIC FRAMEWORK FOR MINLP WITH SEPARABLE NON-CONVEXITY

CLAUDIA D'AMBROSIO*, JON LEE†, AND ANDREAS WÄCHTER‡

Abstract. We present an algorithm for Mixed-Integer Nonlinear Programming (MINLP) problems in which the non-convexity in the objective and constraint functions is manifested as the sum of non-convex univariate functions. We employ a lower bounding convex MINLP relaxation obtained by approximating each non-convex function with a piecewise-convex underestimator that is repeatedly refined. The algorithm is implemented at the level of a modeling language. Favorable numerical results are presented.

Key words. Mixed-integer nonlinear programming, global optimization, spatial branch-and-bound, separable, non-convex.

AMS(MOS) subject classifications. 65K05, 90C11, 90C26, 90C30.

1. Introduction. The global solution of practical instances of Mixed-Integer NonLinear Programming (MINLP) problems has been considered for some decades. Over a considerable period of time, technology for the global optimization of convex MINLP (i.e., the continuous relaxation of the problem is a convex program) had matured (see, for example, [8, 17, 9, 3]), and rather recently there has been considerable success in the realm of global optimization of non-convex MINLP (see, for example, [18, 16, 13, 2]).

Global optimization algorithms, e.g., spatial branch-and-bound approaches like those implemented in codes like BARON [18] and COUENNE [2], have had substantial success in tackling complicated, but generally small scale, non-convex MINLPs (i.e., mixed-integer nonlinear programs having non-convex continuous relaxations). Because they are aimed at a rather general class of problems, the possibility remains that larger instances from a simpler class may be amenable to a simpler approach.

We focus on MINLPs for which the non-convexity in the objective and constraint functions is manifested as the sum of non-convex univariate functions. There are many problems that are already in such a form, or can be brought into such a form via some simple substitutions. In fact, the first step in spatial branch-and-bound is to bring problems into nearly such a form. For our purposes, we assume that the model already has this form. We have developed a simple algorithm, implemented at the level of a modeling language (in our case AMPL, see [10]), to attack such separable problems. First, we identify subintervals of convexity and concavity for the univariate functions using external calls to MATLAB [14]. With such an

*Department of ECSS, University of Bologna, Italy (c.dambrosio@unibo.it).

†Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109, U.S.A. (jonxlee@umich.edu).

‡IBM T.J. Watson Research Center, NY 10598, U.S.A. (andreasw@us.ibm.com).

identification at hand, we develop a convex MINLP relaxation of the problem. Our convex MINLP relaxation differs from those typically employed in spatial branch-and-bound; rather than relaxing the graph of a univariate function on an interval to an enclosing polygon, we work on each subinterval of convexity and concavity separately, using linear relaxation on only the “concave side” of each function on the subintervals. The subintervals are glued together using binary variables. Next, we employ ideas of spatial branch-and-bound, but rather than branching, we repeatedly refine our convex MINLP relaxation by modifying it at the modeling level. We attack our convex MINLP relaxation, to get lower bounds on the global minimum, using the code **BONMIN** [3, 4] as a black-box convex MINLP solver. Finally, by fixing the integer variables in the original non-convex MINLP, and then locally solving the associated non-convex NLP restriction, we get an upper bound on the global minimum, using the code **IPOPT** [19]. We use the solutions found by **BONMIN** and **IPOPT** to guide our choice of further refinements.

We implemented our framework using the modeling language **AMPL**. In order to obtain all of the information necessary for the execution of the algorithm, external software, specifically the tool for high-level computational analysis **MATLAB**, the convex MINLP solver **BONMIN**, and the NLP solver **IPOPT**, are called directly from the **AMPL** environment. A detailed description of the entire algorithmic framework, together with a proof of its convergence, is provided in 2.

We present computational results in 3. Some of the instances arise from specific applications; in particular, Uncapacitated Facility Location problems, Hydro Unit Commitment and Scheduling problems, and Nonlinear Continuous Knapsack problems. We also present computational results on selected instances of **GLOBALLib** and **MINLPLib**. We have had significant success in our preliminary computational experiments. In particular, we see very few major iterations occurring, with most of the time being spent in the solution of a small number of convex MINLPs. As we had hoped, our method does particularly well on problems for which the non-convexity is naturally separable. An advantage of our approach is that it can be implemented easily using existing software components and that further advances in technology for convex MINLP will immediately give us a proportional benefit.

Finally, we note that a preliminary shorter version of the present paper appeared as [7].

2. Our algorithmic framework. We focus now on MINLPs, where the non-convexity in the objective and constraint functions is manifested as the sum of non-convex univariate functions. Without loss of generality, we take them to be of the form

$$\begin{aligned}
& \min \sum_{j \in N} C_j x_j \\
& \text{subject to} \\
& f(x) \leq 0 ; \\
& r_i(x) + \sum_{k \in H(i)} g_{ik}(x_k) \leq 0 , \forall i \in M ; \\
& L_j \leq x_j \leq U_j , \forall j \in N ; \\
& x_j \text{ integer}, \forall j \in I ,
\end{aligned} \tag{\mathcal{P}}$$

where $N := \{1, 2, \dots, n\}$, $f : \mathbb{R}^n \rightarrow \mathbb{R}^p$ and $r_i : \mathbb{R}^n \rightarrow \mathbb{R} \forall i \in M$, are convex functions, $H(i) \subseteq N \forall i \in M$, the $g_{ik} : \mathbb{R} \rightarrow \mathbb{R}$ are non-convex univariate function $\forall i \in M$, and $I \subseteq N$. Letting $H := \cup_{i \in M} H(i)$, we can take each L_j and U_j to be finite or infinite for $j \in N \setminus H$, but for $j \in H$ we assume that these are finite bounds.

We assume that the problem functions are sufficiently smooth (e.g., twice continuously differentiable) with the exception that we allow the univariate g_{ik} to be continuous functions defined piecewise by sufficiently smooth functions over a finite set of subintervals of $[L_k, U_k]$. Without loss of generality, we have taken the objective function as linear and all of the constraints to be inequalities, and further of the less-than-or-equal variety. Linear equality constraints could be included directly in this formulation, while we assume that nonlinear equalities have been split into two inequality constraints.

Our approach is an iterative technique based on three fundamental ingredients:

- A reformulation method with which we obtain a convex MINLP relaxation \mathcal{Q} of the original problem \mathcal{P} . Solving the convex MINLP relaxation \mathcal{Q} , we obtain a lower bound of our original problem \mathcal{P} ;
- A non-convex NLP restriction \mathcal{R} of the original MINLP problem \mathcal{P} obtained by fixing the variables within the set $\{x_j : j \in I\}$. Locally solving the non-convex NLP restriction \mathcal{R} , we obtain an upper bound of our original problem \mathcal{P} ;
- A refinement technique aimed at improving, at each iteration, the quality of the lower bound obtained by solving the convex MINLP relaxation \mathcal{Q} .

The main idea of our algorithmic framework is to iteratively solve a lower-bounding relaxation \mathcal{Q} and an upper-bounding restriction \mathcal{R} so that, in case the value of the upper and the lower bound are the same, the global optimality of the solution found is proven; otherwise we make a refinement to the lower-bounding relaxation \mathcal{Q} . At each iteration, we seek to decrease the gap between the lower and the upper bound, and hopefully, before too long, the gap will be within a tolerance value, or the lower bounding solution is deemed to be sufficiently feasible for the original problem. In these cases, or in the case a time/iteration limit is reached, the algorithm stops. If the gap is closed, we have found a global optimum, otherwise we have a heuristic solution (provided that the upper bound is not $+\infty$). The lower-bounding relaxation \mathcal{Q} is a convex relaxation of the original non-

convex MINLP problem, obtained by approximating the concave part of the non-convex univariate functions using piecewise linear approximation. The novelty in this part of the algorithmic framework is the new formulation of the convex relaxation: The function is approximated only where it is concave, while the convex parts of the functions are not approximated, but taken as they are. The convex relaxation proposed is described in details in 2.1. The upper-bounding restriction \mathcal{R} , described in 2.2, is obtained simply by fixing the variables with integrality constraints. The refinement technique consists of adding one or more breakpoints where needed, i.e., where the approximation of the non-convex function is bad and the solution of the lower-bounding problem lies. Refinement strategies are described in 2.3, and once the ingredients of the algorithmic framework are described in detail, we give a pseudo-code description of our algorithmic framework (see 2.4). Here, we also discuss some considerations about the general framework and the similarities and differences with popular global optimization methods. Theoretical convergence guarantees are discussed in 2.5. In 3, computational experiments are presented, detailing the performance of the algorithm and comparing the approach to other methods.

2.1. The lower-bounding convex MINLP relaxation \mathcal{Q} . To obtain our convex MINLP relaxation \mathcal{Q} of the MINLP problem \mathcal{P} , we need to locate the subintervals of the domain of each univariate function g_i for which the function is uniformly convex or concave. For simplicity of notation, rather than refer to the constraint $r_i(x) + \sum_{k \in H(i)} g_{ik}(x_k) \leq 0$, we consider a term of the form $g(x_k) := g_{ik}(x_k)$, where $g : \mathbb{R} \rightarrow \mathbb{R}$ is a univariate non-convex function of x_k , for some k ($1 \leq k \leq n$).

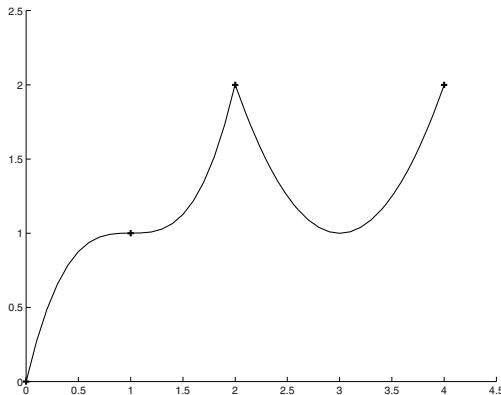
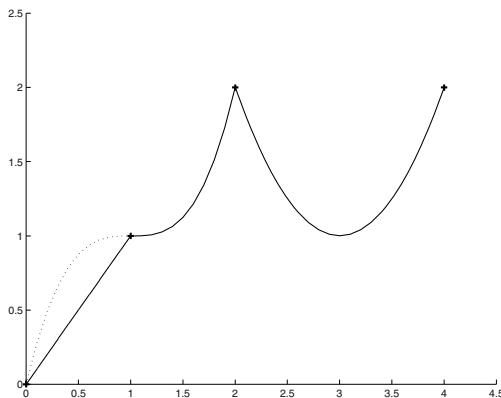
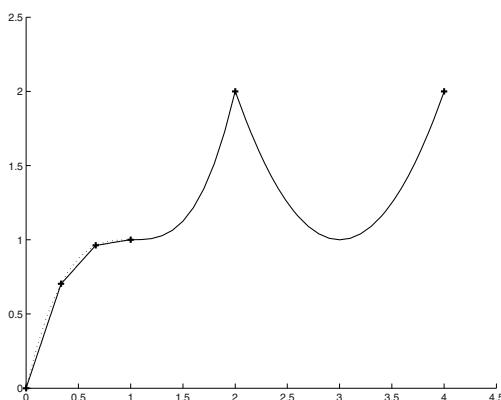
We want to explicitly view each such g as a piecewise-defined function, where on each piece the function is either convex or concave. This feature also allows us to handle functions that are already piecewise defined by the modeler. In practice, for each non-convex function g , we compute the points at which the convexity/concavity may change, i.e., the zeros of the second derivative of g , using MATLAB. In case a function g is naturally piecewise defined, we are essentially refining the piecewise definition of it in such a way that the convexity/concavity is uniform on each piece.

EXAMPLE 1. Consider the piecewise-defined univariate function

$$g(x_k) := \begin{cases} 1 + (x_k - 1)^3, & \text{for } 0 \leq x_k \leq 2; \\ 1 + (x_k - 3)^2, & \text{for } 2 \leq x_k \leq 4, \end{cases}$$

depicted in FIG. 1. In addition to the breakpoints $x_k = 0, 2, 4$ of the definition of g , the convexity/concavity changes at $x_k = 1$, so by utilizing an additional breakpoint at $x_k = 1$ the convexity/concavity is now uniform on each piece.

Now, on each concave piece we can use a secant approximation to give a piecewise-convex lower approximation of g .

FIG. 1. *A piecewise-defined univariate function.*FIG. 2. *A piecewise-convex lower approximation.*FIG. 3. *Improved piecewise-convex lower approximation.*

EXAMPLE 1, continued. *Relative to $g(x_k)$ of EXAMPLE 1, we have the piecewise-convex lower approximation*

$$\underline{g}(x_k) := \begin{cases} x_k, & \text{for } 0 \leq x_k \leq 1; \\ 1 + (x_k - 1)^3, & \text{for } 1 \leq x_k \leq 2; \\ 1 + (x_k - 3)^2, & \text{for } 2 \leq x_k \leq 4, \end{cases}$$

depicted in FIG. 2.

We can obtain a better lower bound by refining the piecewise-linear lower approximation on the concave pieces. We let

$$L_k =: P_0 < P_1 < \cdots < P_{\bar{p}} := U_k \quad (2.1)$$

be the ordered breakpoints at which the convexity/concavity of g changes, including, in the case of piecewise definition of g , the points at which the definition of g changes. We define:

- $[P_{p-1}, P_p] :=$ the p -th subinterval of the domain of g ($p \in \{1 \dots \bar{p}\}$);
- $\check{H} :=$ the set of indices of subintervals on which g is convex;
- $\hat{H} :=$ the set of indices of subintervals on which g is concave.

On the concave intervals, we will allow further breakpoints. We let B_p be the ordered set of breakpoints for the concave interval indexed by $p \in \hat{H}$. We denote these breakpoints as

$$P_{p-1} =: X_{p,1} < X_{p,2} < \cdots < X_{p,|B_p|} := P_p,$$

and in our relaxation we will view g as lower bounded by the piecewise-linear function that has value $g(X_{p,j})$ at the breakpoints $X_{p,j}$, and is otherwise linear between these breakpoints.

EXAMPLE 1, continued again. *Utilizing further breakpoints, for example at $x_k = 1/3$ and $x_k = 2/3$, we can improve the piecewise-convex lower approximation to*

$$\underline{g}(x_k) := \begin{cases} \frac{19}{9}x_k, & \text{for } 0 \leq x_k \leq \frac{1}{3}; \\ \frac{19}{27} + \frac{7}{9}(x_k - \frac{1}{3}), & \text{for } \frac{1}{3} \leq x_k \leq \frac{2}{3}; \\ \frac{26}{27} + \frac{1}{9}(x_k - \frac{2}{3}), & \text{for } \frac{2}{3} \leq x_k \leq 1; \\ 1 + (x_k - 1)^3, & \text{for } 1 \leq x_k \leq 2; \\ 1 + (x_k - 3)^2, & \text{for } 2 \leq x_k \leq 4, \end{cases}$$

depicted in FIG. 3.

Next, we define further variables to manage our convexification of g on its domain:

$z_p :=$ a binary variable indicating if $x_k \geq P_p$ ($p = 1, \dots, \bar{p} - 1$);

$\delta_p :=$ a continuous variable assuming a positive value iff $x_k \geq P_{p-1}$ ($p = 1, \dots, \bar{p}$);

$\alpha_{p,b} :=$ weight of breakpoint b in the piecewise-linear approximation of the interval indexed by p ($p \in \check{H}$, $b \in B_p$).

In the convex relaxation of the original MINLP \mathcal{P} , we substitute each univariate non-convex term $g(x_k)$ with

$$\sum_{p \in \check{H}} g(P_{p-1} + \delta_p) + \sum_{p \in \hat{H}} \sum_{b \in B_p} g(X_{p,b}) \alpha_{p,b} - \sum_{p=1}^{\bar{p}-1} g(P_p) , \quad (2.2)$$

and we include the following set of new constraints:

$$P_0 + \sum_{p=1}^{\bar{p}} \delta_p - x_k = 0 ; \quad (2.3)$$

$$\delta_p - (P_p - P_{p-1}) z_p \geq 0 , \quad \forall p \in \check{H} \cup \hat{H} ; \quad (2.4)$$

$$\delta_p - (P_p - P_{p-1}) z_{p-1} \leq 0 , \quad \forall p \in \check{H} \cup \hat{H} ; \quad (2.5)$$

$$P_{p-1} + \delta_p - \sum_{b \in B_p} X_{p,b} \alpha_{p,b} = 0 , \quad \forall p \in \hat{H} ; \quad (2.6)$$

$$\sum_{b \in B_p} \alpha_{p,b} = 1 , \quad \forall p \in \hat{H} ; \quad (2.7)$$

$$\{\alpha_{p,b} : b \in B_p\} := \text{SOS2} , \quad \forall p \in \hat{H} , \quad (2.8)$$

with two dummy variables $z_0 := 1$ and $z_{\bar{p}} := 0$.

Constraints (2.3–2.5) together with the integrality of the z variables ensure that, given an x_k value, say $x_k^* \in [P_{p^*-1}, P_{p^*}]$:

$$\delta_p = \begin{cases} P_p - P_{p-1} , & \text{if } 1 \leq p \leq p^* - 1 ; \\ x_k^* - P_{p-1} , & \text{if } p = p^* ; \\ 0 , & \text{otherwise.} \end{cases}$$

Constraints (2.6–2.8) ensure that, for each concave interval, the convex combination of the breakpoints is correctly computed. Finally, (2.2) approximates the original non-convex univariate function $g(x_k)$. Each single term of the first and the second summations, using the definition of δ_p , reduces, respectively, to

$$g(P_{p-1} + \delta_p) = \begin{cases} g(P_p) , & \text{if } p \in \{1, \dots, p^* - 1\} ; \\ g(x_k^*) , & \text{if } p = p^* ; \\ g(P_{p-1}) , & \text{if } p \in \{p^* + 1, \dots, \bar{p}\} , \end{cases}$$

and

$$\sum_{b \in B_p} g(X_{p,b}) \alpha_{p,b} = \begin{cases} g(P_p) , & \text{if } p \in \{1, \dots, p^* - 1\} ; \\ \sum_{b \in B_{p^*}} g(X_{p^*,b}) \alpha_{p^*,b} , & \text{if } p = p^* ; \\ g(P_{p-1}) , & \text{if } p \in \{p^* + 1, \dots, \bar{p}\} , \end{cases}$$

reducing expression (2.2) to

$$\sum_{p=1}^{p^*-1} g(P_p) + \gamma + \sum_{p=p^*+1}^{\bar{p}} g(P_{p-1}) - \sum_{p=1}^{\bar{p}-1} g(P_p) = \gamma ,$$

where

$$\gamma = \begin{cases} g(x_k^*) , & \text{if } p^* \in \check{H} ; \\ \sum_{b \in B_{p^*}} g(X_{p^*, b}) \alpha_{p^* b} , & \text{if } p^* \in \hat{H} . \end{cases}$$

So, if x_k^* is in a subinterval on which g is convex, then the approximation (2.2) is exact; while if x_k^* is in a subinterval on which g is concave, then the approximation is a piecewise-linear underestimation of g .

Constraints (2.8) define $|\hat{H}|$ Special Ordered Sets of Type 2 (SOS2), i.e., ordered sets of positive variables among which at most 2 can assume a non-zero value, and, in this case, they must be consecutive (see Beale and Tomlin [1]). Unfortunately, at the moment, convex MINLP solvers do not typically handle SOS2 like most MILP solvers do (also defining special-purpose branching strategies). For this reason, we substitute constraints (2.8), $\forall p \in \hat{H}$, with new binary variables $y_{p,b}$, with $b \in \{1, \dots, |B_p| - 1\}$, and constraints:

$$\alpha_{p,b} \leq y_{p,b-1} + y_{p,b} \quad \forall b \in B_p ; \quad (2.8.a)$$

$$\sum_{b=1}^{|B_p|-1} y_{p,b} = 1 , \quad (2.8.b)$$

with dummy values $y_{p,0} = y_{p,|B_p|} = 0$. In the future, when convex MINLP solvers will handle the definition of SOS2, variables y and constraints (2.8.a–b) would be not necessary.

It is important to note that if we utilized a very large number of breakpoints at the start, solving the resulting convex MINLP \mathcal{Q} would mean essentially solving globally the original MINLP \mathcal{P} up to some pre-determined tolerance related to the density of the breakpoints. But of course such a convex MINLP \mathcal{Q} would be too hard to be solved in practice. With our algorithmic framework, we dynamically seek a significantly smaller convex MINLP \mathcal{Q} , thus generally more easily solvable, which we can use to guide the non-convex NLP restriction \mathcal{R} to a good local solution, eventually settling on and proving global optimality of such a solution to the original MINLP \mathcal{P} .

2.2. The upper-bounding non-convex NLP restriction \mathcal{R} . Given a solution \underline{x} of the convex MINLP relaxation \mathcal{Q} , the upper-bounding restriction \mathcal{R} is defined as the non-convex NLP:

$$\begin{aligned} \min \quad & \sum_{j \in N} C_j x_j \\ \text{subject to} \quad & f(x) \leq 0 ; \\ & r_i(x) + \sum_{k \in H(i)} g_{ik}(x_k) \leq 0 , \quad \forall i \in M ; \\ & L_j \leq x_j \leq U_j , \quad \forall j \in N ; \\ & x_j = \underline{x}_j , \quad \forall j \in I . \end{aligned} \quad (\mathcal{R})$$

A solution of this non-convex NLP \mathcal{R} is a heuristic solution of the non-convex MINLP problem \mathcal{P} for two reasons: (i) the integer variables

x_j , $j \in I$, might not be fixed to globally optimal values; (ii) the NLP \mathcal{R} is non-convex, and so even if the integer variables x_j , $j \in I$, are fixed to globally optimal values, the NLP solver may only find a local optimum of the non-convex NLP \mathcal{R} or even fail to find a feasible point. This consideration emphasizes the importance of the lower-bounding relaxation \mathcal{Q} for the guarantee of global optimality. The upper-bounding problem resolution could be seen as a “verification phase” in which a solution of the convex MINLP relaxation \mathcal{Q} is tested to be really feasible for the non-convex MINLP \mathcal{P} . To emphasize this, the NLP solver for \mathcal{R} is given the solution of the convex MINLP relaxation as starting point.

2.3. The refinement technique. At the end of each iteration, we have two solutions: \underline{x} , the solution of the lower-bounding convex MINLP relaxation \mathcal{Q} , and \bar{x} , the solution of the upper-bounding non-convex NLP restriction \mathcal{R} ; in case we cannot find a solution of \mathcal{R} , e.g., if \mathcal{R} is infeasible, then no \bar{x} is available. If $\sum_{j \in N} C_j \underline{x}_j = \sum_{j \in N} C_j \bar{x}_j$ within a certain tolerance, or if \underline{x} is sufficiently feasible for the original constraints, we return to the user as solution the point \bar{x} or \underline{x} , respectively. Otherwise, in order to continue, we want to refine the approximation of the lower-bounding convex MINLP relaxation \mathcal{Q} by adding further breakpoints. We employed two strategies:

- *Based on the lower-bounding problem solution \underline{x} : For each $i \in M$ and $k \in H(i)$, if \underline{x}_k lies in a concave interval of g_{ik} , add \underline{x}_k as a breakpoint for the relaxation of g_{ik} .*

This procedure drives the convergence of the overall method since it makes sure that the lower bounding problem becomes eventually a sufficiently accurate approximation of the original problem in the neighborhood of the global solution. Since adding a breakpoint increases the size of the convex MINLP relaxation, in practice we do not add such a new breakpoint if it would be within some small tolerance of an existing breakpoint for g_{ik} .

- *Based on the upper-bounding problem solution \bar{x} : For each $i \in M$ and $k \in H(i)$, if \bar{x}_k lies in a concave interval of g_{ik} , add \bar{x}_k as a breakpoint for the relaxation of g_{ik} .*

The motivation behind this option is to accelerate the convergence of the method. If the solution found by the upper-bounding problem is indeed the global solution, the relaxation should eventually be exact at this point to prove its optimality. Again, to keep the size of the relaxation MINLP manageable, breakpoints are only added if they are not too close to existing ones.

We found that these strategies work well together. Hence, at each major iteration, we add a breakpoint in each concave interval where \underline{x} lies in order to converge and one where \bar{x} lies to speed up the convergence.

2.4. The algorithmic framework. Algorithm 1 details our SC-MINLP (Sequential Convex MINLP) Algorithm, while [Figure 4](#) depicts,

at a high level, how we designed our implementation of it. For an optimization problem, $\text{val}(\cdot)$ refers to its optimal objective value.

Algorithm 1 : SC-MINLP (Sequential Convex MINLP) Algorithm

```

Choose tolerances  $\varepsilon, \varepsilon_{\text{feas}} > 0$ ; initialize  $LB := -\infty$ ;  $UB := +\infty$ ;
Find  $P_p^i, \hat{H}^i, \check{H}^i, X_{pb}^i$  ( $\forall i \in M, p \in \{1 \dots \bar{p}^i\}, b \in B_p^i$ ).
repeat
  Solve the convex MINLP relaxation  $\mathcal{Q}$  of the original problem  $\mathcal{P}$  to
  obtain  $\underline{x}$ ;
  if ( $\underline{x}$  is feasible for the original problem  $\mathcal{P}$  (within tolerance  $\varepsilon_{\text{feas}}$ ))
  then
    return  $\underline{x}$ 
  end if
  if ( $\text{val}(\mathcal{Q}) > LB$ ) then
     $LB := \text{val}(\mathcal{Q})$ ;
  end if
  Solve the non-convex NLP restriction  $\mathcal{R}$  of the original problem  $\mathcal{P}$  to
  obtain  $\bar{x}$ ;
  if (solution  $\bar{x}$  could be computed and  $\text{val}(\mathcal{R}) < UB$ ) then
     $UB := \text{val}(\mathcal{R})$ ;  $x_{UB} := \bar{x}$ 
  end if
  if ( $UB - LB > \varepsilon$ ) then
    Update  $B_p^i, X_{pb}^i$  ;
  end if
until ( $(UB - LB \leq \varepsilon)$  or (time or iteration limited exceeded))
return the current best solution  $x_{UB}$ 

```

At each iteration, the lower-bounding MINLP relaxation \mathcal{Q} and the upper-bounding NLP restriction \mathcal{R} are redefined: What changes in \mathcal{Q} are the sets of breakpoints that refine the piecewise-linear approximation of concave parts of the non-convex functions. At each iteration, the number of breakpoint used increases, and so does the accuracy of the approximation. What may change in \mathcal{R} are the values of the fixed integer variables \underline{x}_j , $j \in I$. Moreover, what changes is the starting point given to the NLP solver, derived from an optimal solution of the lower-bounding MINLP relaxation \mathcal{Q} .

Our algorithmic framework bears comparison with spatial branch-and-bound, a successful technique in global optimization. In particular:

- during the refining phase, the parts in which the approximation is bad are discovered and the approximation is improved there, but we do it by adding one or more breakpoints instead of branching on a continuous variable as in spatial branch-and-bound;
- like spatial branch-and-bound, our approach is a rigorous global-optimization algorithm rather than a heuristic;

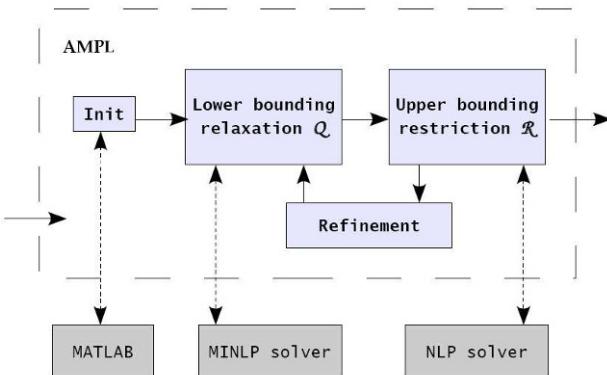


FIG. 4. The SC-MINLP (Sequential Convex MINLP) framework.

- unlike spatial branch-and-bound, our approach does not utilize an expression tree; it works directly on the broad class of separable non-convex MINLPs of the form \mathcal{P} , and of course problems that can be put in such a form;
- unlike standard implementations of spatial branch-and-bound methods, we can directly keep multivariate convex functions in our relaxation instead of using linear approximations;
- unlike spatial branch-and-bound, our method can be effectively implemented at the modeling-language level.

2.5. Convergence analysis. For the theoretical convergence analysis of Algorithm 1, we make the following assumptions, denoting by l the iteration counter for the **repeat** loop.

- A1. The functions $f(x)$ and $r_i(x)$ are continuous, and the univariate functions g_{ik} in (\mathcal{P}) are uniformly Lipschitz-continuous with a bounded Lipschitz constant L_g .
- A2. The set of breakpoints (2.1) are correctly identified.
- A3. The problem \mathcal{P} has a feasible point. Hence, for each l , the relaxation Q^l is feasible, and we assume its (globally) optimal solution \underline{x}^l is computed.
- A4. The refinement technique described in Section 2.3 adds a breakpoint for every lower-bounding problem solution \underline{x}^l , even if it is very close to an existing breakpoint.
- A5. The feasibility tolerance $\varepsilon_{\text{feas}}$ and the optimality gap tolerance ε are both chosen to be zero, and no iteration limit is set.

THEOREM 2.1. *Under assumptions A1-A5, Algorithm 1 either terminates at a global solution of the original problem \mathcal{P} , or each limit point of the sequence $\{\underline{x}^l\}_{l=1}^\infty$ is a global solution of \mathcal{P} .*

Proof. By construction, \mathcal{Q}^l is always a relaxation of \mathcal{P} , and hence $\text{val}(\mathcal{Q}^l)$ is always less than or equal to the value $\text{val}(\mathcal{P})$ of the objective function at the global solution to \mathcal{P} .

If the algorithm terminates in a finite number of iterations, it either returns an iterate \underline{x}_l that is feasible for \mathcal{P} (after the feasibility test for \mathcal{P}) and therefore a global solution for \mathcal{P} , or it returns a best upper bounding solution x_{UB} , which as a solution for \mathcal{R} is feasible for \mathcal{P} and has the same value as any global solution for \mathcal{P} (since $UB = LB$).

In the case that the algorithm generates an infinite sequence $\{\underline{x}^l\}$ of iterates, let \underline{x}^* be a limit point of this sequence, i.e., there exists a subsequence $\{\underline{x}^{\tilde{l}}\}_{\tilde{l}=1}^\infty$ of $\{\underline{x}^l\}_{l=1}^\infty$ converging to \underline{x}^* . As a solution of $\mathcal{Q}^{\tilde{l}}$, each $\underline{x}^{\tilde{l}}$ satisfies the constraints of \mathcal{P} , except for

$$r_i(x) + \sum_{k \in H(i)} g_{ik}(x_k) \leq 0, \quad \forall i \in M, \quad (2.9)$$

because the “ $g_{ik}(x_k)$ ” terms are replaced by a piecewise convex relaxation, see (2.2). We denote the values of their approximation by $\tilde{g}_{ik}^{\tilde{l}}(x_k)$. Then, the solutions of $\mathcal{Q}^{\tilde{l}}$ satisfy

$$r_i(\underline{x}^{\tilde{l}}) + \sum_{k \in H(i)} \tilde{g}_{ik}^{\tilde{l}}(\underline{x}_k^{\tilde{l}}) \leq 0, \quad \forall i \in M. \quad (2.10)$$

Now choose \tilde{l}_1, \tilde{l}_2 with $\tilde{l}_2 > \tilde{l}_1$. Because the approximation $\tilde{g}_{ik}^{\tilde{l}_2}(x_k)$ is defined to coincide with the convex parts of $g_{ik}(x_k)$ and is otherwise a linear interpolation between breakpoints (note that $\underline{x}_k^{\tilde{l}_1}$ is a breakpoint for $\mathcal{Q}^{\tilde{l}_2}$ if $\underline{x}_k^{\tilde{l}_1}$ is in an interval where $g_{ik}(x_k)$ is concave), the Lipschitz-continuity of the $g_{ik}(x_k)$ gives us

$$\tilde{g}_{ik}^{\tilde{l}_2}(\underline{x}_k^{\tilde{l}_2}) \geq g_{ik}(\underline{x}_k^{\tilde{l}_1}) - L_g |\underline{x}_k^{\tilde{l}_2} - \underline{x}_k^{\tilde{l}_1}|.$$

Together with (2.10) we therefore obtain

$$\begin{aligned} r_i(\underline{x}^{\tilde{l}_2}) + \sum_{k \in H(i)} g_{ik}(\underline{x}_k^{\tilde{l}_1}) &\leq r_i(\underline{x}^{\tilde{l}_2}) + \sum_{k \in H(i)} \tilde{g}_{ik}^{\tilde{l}_2}(\underline{x}_k^{\tilde{l}_2}) + L_g \sum_{k \in H(i)} |\underline{x}_k^{\tilde{l}_2} - \underline{x}_k^{\tilde{l}_1}| \\ &\leq L_g \sum_{k \in H(i)} |\underline{x}_k^{\tilde{l}_2} - \underline{x}_k^{\tilde{l}_1}| \end{aligned}$$

for all $i \in M$. Because \tilde{l}_1, \tilde{l}_2 with $\tilde{l}_2 > \tilde{l}_1$ have been chosen arbitrarily, taking the limit as $\tilde{l}_1, \tilde{l}_2 \rightarrow \infty$ and using the continuity of r shows that \underline{x}^* satisfies (2.9). The continuity of the remaining constraints in \mathcal{P} ensure that \underline{x}^* is feasible for \mathcal{P} . Because $\text{val}(\mathcal{Q}^{\tilde{l}}) \leq \text{val}(\mathcal{P})$ for all \tilde{l} , we finally obtain $\text{val}(\mathcal{Q}^*) \leq \text{val}(\mathcal{P})$, so that \underline{x}^* must be a global solution of \mathcal{P} . \square

3. Computational results. We implemented our algorithmic framework as an AMPL script, and we used MATLAB as a tool for numerical convexity analysis, BONMIN as our convex MINLP solver, and IPOPT as our NLP solver.

We used MATLAB to detect the subintervals of convexity and concavity for the non-convex univariate functions in the model. In particular, MATLAB reads a text file generated by the AMPL script, containing the constraints with univariate non-convex functions, together with the names and bounds of the independent variables. With this information, using the Symbolic Math Toolbox, MATLAB first computes the formula for the second derivative of each univariate non-convex function, and then computes its zeros to split the function into subintervals of convexity and concavity. The zeros are computed in the following manner. The second derivative is evaluated at 100 evenly-spaced points on the interval defined by the variable's lower and upper bounds. Then, between points where there is a change in the sign of the second derivative, another 100 evenly-spaced points are evaluated. Finally, the precise location of each breakpoint is computed using the MATLAB function “fzero”. For each univariate non-convex function, we use MATLAB to return the number of subintervals, the breakpoints, and associated function values in a text file which is read by the AMPL script.

In this section we present computational results for four problem categories. The tests were executed on a single processor of an Intel Core2 CPU 6600, 2.40 GHz with 1.94 GB of RAM, using a time limit of 2 hours per instance. The relative optimality gap and feasibility tolerance used for all the experiments is 10^{-4} , and we do not add a breakpoint if it would be within 10^{-5} of an existing breakpoint.

For each set of problems, we describe the non-convex MINLP model \mathcal{P} . Two tables with computational results exhibit the behavior of our algorithm on some instances of each problem class. The first table presents the iterations of our SC-MINLP Algorithm, with the columns labeled as follows:

- instance: the instance name;
- var/int/cons: the total number of variables, the number of integer variables, and the number of constraints in the convex relaxation \mathcal{Q} ;
- iter #: the iteration count;
- LB: the value of the lower bound;
- UB: the value of the upper bound;
- int change: indicated whether the integer variables in the lower bounding solution \underline{x} are different compared to the previous iteration;
- time MINLP: the CPU time needed to solve the convex MINLP relaxation \mathcal{Q} to optimality (in seconds);
- # br added: the number of breakpoints added at the end of the previous iteration.

The second table presents comparisons of our SC-MINLP Algorithm with COUENNE and BONMIN. COUENNE is an open-source Branch-and-Bound algorithm aimed at the global solution of MINLP problems [2, 6]. It is an exact method for the problems we address in this paper. BONMIN is an open-source code for solving general MINLP problems [3, 4], but it is an exact method only for convex MINLPs. Here, BONMIN's nonlinear branch-and-bound option was chosen. When used for solving non-convex MINLPs, the solution returned is not guaranteed to be a global optimum. However, a few heuristic options are available in BONMIN, specifically designed to treat non-convex MINLPs. Here, we use the option that allows solving the root node with a user-specified number of different randomly-chosen starting points, continuing with the best solution found. This heuristic use of BONMIN is in contrast to its use in SC-MINLP, where BONMIN is employed only for the solution of the *convex* MINLP relaxation \mathcal{Q} .

The columns in the second table have the following meaning:

- instance: the instance name;
- var/int/cons: the total number of variables, the number of integer variables, and the number of constraints;
- for each approach, in particular SC-MINLP, COUENNE, BONMIN 1, BONMIN 50, we report:
 - time (LB): the CPU time (or the value of the lower bound (in parentheses) if the time limit is reached);
 - UB: the value of the upper bound.

BONMIN 1 and BONMIN 50 both refer to the use of BONMIN, but they differ in the number of multiple solutions of the root node; in the first case, the root node is solved just once, while in the second case, 50 randomly-generated starting points are given to the root-node NLP solver. If BONMIN reached the time limit, we do not report the lower bound because BONMIN cannot determine a valid lower bound for a non-convex problem.

3.1. Uncapacitated Facility Location (UFL) problem. The UFL application is presented in [12]. The set of customers is denoted with T and the set of facilities is denoted with K (w_{kt} is the fraction of demand of customer t satisfied by facility k for each $t \in T, k \in K$). Univariate non-convexity in the model arises due to nonlinear shipment costs. The UFL model formulation is as follows:

$$\begin{aligned}
 & \min \sum_{k \in K} C_k y_k + \sum_{t \in T} v_t \\
 & \text{subject to} \\
 & v_t \geq -\sum_{k \in K} S_{kt} s_{kt}, \quad \forall t \in T; \\
 & s_{kt} \leq g_{kt}(w_{kt}), \quad \forall t \in T; \\
 & w_{kt} \leq y_k, \quad \forall t \in T, k \in K; \\
 & \sum_{k \in K} w_{kt} = 1, \quad \forall t \in T; \\
 & w_{kt} \geq 0, \quad \forall t \in T, k \in K; \\
 & y_k \in \{0, 1\}, \quad \forall k \in K.
 \end{aligned}$$

[Figure 5](#) depicts the three different nonlinear functions $g_{kt}(w_{kt})$ that were used for the computational results presented in [Tables 1](#) and [2](#). The dashed line depicts the non-convex function, while the solid line indicates the initial piecewise-convex underestimator. Note that the third function was intentionally designed to be pathological and challenging for SC-MINLP. The remaining problem data was randomly generated.

In [Table 1](#) the performance of SC-MINLP is shown. For the first instance, the global optimum is found at the first iteration, but 4 more iteration are needed to prove global optimality. In the second instance, only one iteration is needed. In the third instance, the first feasible solution found is not the global optimum which is found at the third (and last) iteration. [Table 2](#) demonstrates good performance of SC-MINLP. In particular, instance `uf1_1` is solved in about 117 seconds compared to 530 seconds needed by COUENNE, instance `uf1_2` in less than 18 seconds compared to 233 seconds. In instance `uf1_3`, COUENNE performs better than SC-MINLP, but this instance is really quite easy for both algorithms. BONMIN 1 finds solutions to all three instances very quickly, and these solutions turn out to be globally optimal (but note, however, that BONMIN 1 is a heuristic algorithm and no guarantee of the global optimality is given). BONMIN 50 also finds the three global optima, but in non-negligible time (greater than the one needed by SC-MINLP in 2 out of 3 instances).

3.2. Hydro Unit Commitment and Scheduling problem. The Hydro Unit Commitment and Scheduling problem is described in [5]. Univariate non-convexity in the model arises due to the dependence of the power produced by each turbine on the water flow passing through the turbine. The following model was used for the computational results of [Tables 3](#) and [4](#):

$$\begin{aligned}
& \min - \sum_{j \in J} \sum_{t \in T} \left(\Delta t \Pi_t p_{jt} - C_j \tilde{w}_{jt} - (D_j + \Pi_t E_j) \tilde{y}_{jt} \right) \\
& \text{subject to} \\
& v_{\bar{t}} - V_{\bar{t}} = 0 ; \\
& v_t - v_{t-1} - 3600 \Delta t (I_t - \sum_{j \in J} q_{jt} - s_t) = 0 , \quad \forall t \in T ; \\
& q_{jt} - (Q_j^- u_{jt} + Q_j^+ g_{jt}) \geq 0 , \quad \forall j \in J, t \in T ; \\
& q_{jt} - (Q_j^- u_{jt} + Q_j^+ g_{jt}) \leq 0 , \quad \forall j \in J, t \in T ; \\
& \sum_{j \in J} (q_{jt} - q_{j(t-1)}) + \Delta q^- \geq 0 , \quad \forall t \in T ; \\
& \sum_{j \in J} (q_{jt} - q_{j(t-1)}) - \Delta q^+ \leq 0 , \quad \forall t \in T ; \\
& s_t - \sum_{j \in J} (W_j \tilde{w}_{jt} + Y_j \tilde{y}_{jt}) \geq 0 , \quad \forall t \in T ; \\
& \sum_{j \in J} q_{jt} + s_t - \Theta \geq 0 , \quad \forall t \in T ; \\
& g_{jt} - g_{j(t-1)} - (\tilde{w}_{jt} - w_{jt}) = 0 , \quad \forall j \in J, t \in T ; \\
& \tilde{w}_{jt} + w_{jt} \leq 1 , \quad \forall j \in J, t \in T ; \\
& u_{jt} - u_{j(t-1)} - (\tilde{y}_{jt} - y_{jt}) = 0 , \quad \forall j \in J, t \in T ; \\
& \tilde{y}_{jt} + y_{jt} \leq 1 , \quad \forall j \in J, t \in T ; \\
& g_{jt} + u_{kt} \leq 1 , \quad \forall j, k \in J, t \in T ;
\end{aligned}$$

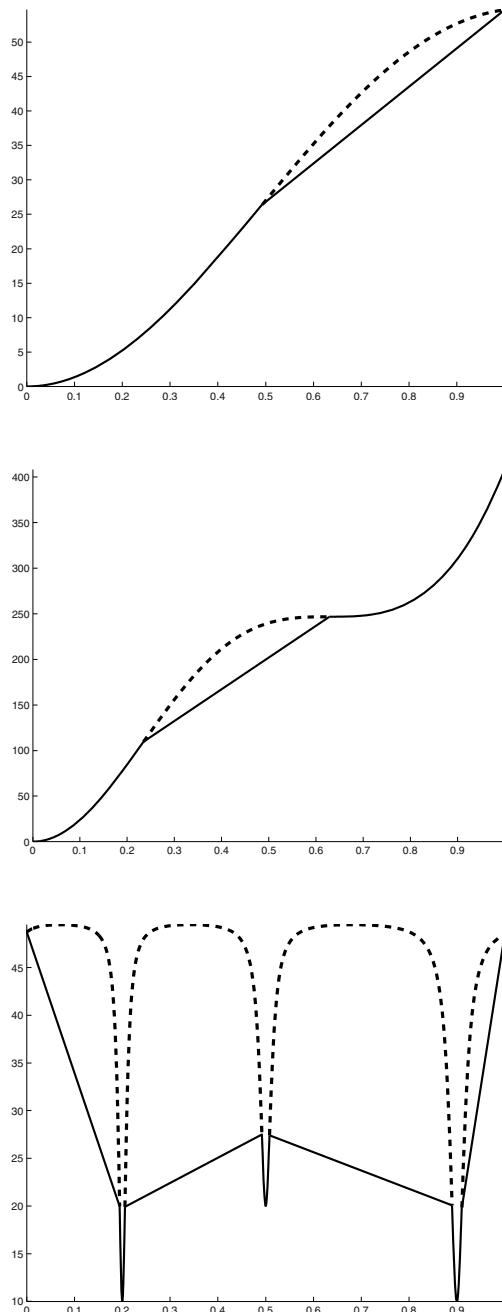

 FIG. 5. UFL: Shapes of $-g_{kt}(w_{kt})$ for the three instances.

TABLE 1
Results for Uncapacitated Facility Location problem.

instance	var/int/cons	iter #	LB	UB	int change	MINLP time	# br added
uf1_1	153/39/228	1	4,122,000	4,330,400	-	1.35	-
	...	2	4,324,780	4,330,400	no	11.84	11
	...	3	4,327,724	4,330,400	no	19.17	5
uf1_2	205/65/254	4	4,328,993	4,330,400	no	30.75	5
	...	5	4,330,070	4,330,400	no	45.42	5
	189/57/264	1	27,516,600	27,516,569	-	4.47	-
uf1_3	79/21/101	1	1,947,883	2,756,890	-	2.25	-
	...	2	2,064,267	2,756,890	no	2.75	2
	87/25/105	3	2,292,743	2,292,777	no	3.06	2

TABLE 2
Results for Uncapacitated Facility Location problem.

instance	var/int/cons original	SC-MINLP		COUENNE		BONMIN 1		BONMIN 50	
		time (LB)	UB	time (LB)	UB	time	UB	time	UB
uf1_1	45/3/48	116.47	4,330.400	529.49	4,330.400	0.32	4,330.400	369.85	4,330.400
uf1_2	45/3/48	17.83	27,516.569	232.85	27,516.569	0.97	27,516.569	144.06	27,516.569
uf1_3	32/2/36	8.44	2,292.777	0.73	2,292.775	3.08	2,292.777	3.13	2,292.775

$$\begin{aligned} \sum_{j \in J} u_{jt} &\leq \bar{n} - 1, \quad \forall t \in T; \\ p_{jt} - \varphi(q_{jt}) &= 0, \quad \forall j \in J, t \in T. \end{aligned}$$

[Figure 6](#) shows the non-convex functions $\varphi(q_{jt})$ used for the three instances. The remaining problem data was chosen according to [5].

Our computational results are reported in [Tables 3](#) and [4](#). We observe good performance of SC-MINLP. It is able to find the global optimum of the three instances within the time limit, but COUENNE does not solve to global optimality any of the instances. Also, BONMIN 1 and BONMIN 50 show good performance. In particular, often a good solution is found in few seconds, and BONMIN 1 finds the global optimum in one case.

3.3. Nonlinear Continuous Knapsack problem. In this subsection, we present results for the Nonlinear Continuous Knapsack problem. This purely continuous problem can be motivated as a continuous resource-allocation problem. A limited resource of amount C (such as advertising dollars) has to be partitioned for different categories $j \in N$ (such as advertisements for different products). The objective function is the overall return from all categories. The non-convexity arises because a small amount of allocation provides only a small return, up to a threshold, when the advertisements are noticed by the consumers and result in substantial sales. On the other hand, at some point, saturation sets in, and an increase of advertisement no longer leads to a significant increase in sales.

Our model is the non-convex NLP problem:

$$\begin{aligned} \min \quad & -\sum_{j \in N} p_j \\ \text{subject to} \quad & p_j - g_j(x_j) \leq 0, \quad \forall j \in N; \\ & \sum_{j \in N} x_j \leq C; \\ & 0 \leq x_j \leq U, \quad \forall j \in N, \end{aligned}$$

where $g_j(x_j) = c_j / (1 + b_j \exp^{-a_j(x_j + d_j)})$.

Note that the sigmoid function $g_j(x_j)$ is increasing, and it is convex up to the inflection point at $x = -d_j + \ln(b_j)/a_j$, whereupon it becomes concave.

The instances were generated randomly. In particular, independently for each $j \in N$, a_j was uniformly generated in the interval $[0.1, 0.2]$, b_j and c_j in the interval $[0, U]$ and d_j in the interval $[-U, 0]$. The name of each instance contains information about the values $|N|$ and C , namely, `nck_` $|N|$ `_C`.

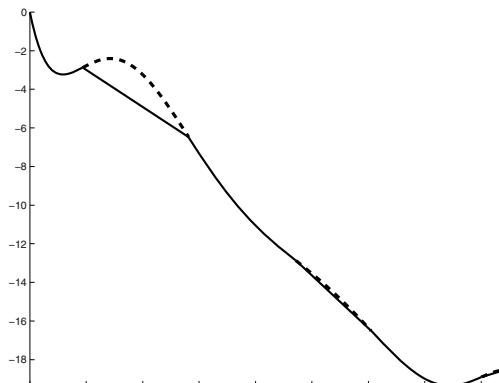
Our computational results are reported in [Tables 5](#) and [6](#). SC-MINLP finds the global optimum for all the 6 instances in less than 3 minutes. COUENNE is able to close the gap for only 2 instances within the time limit. BONMIN 1 and BONMIN 50 terminate quickly, but the global optimum is found only for 1 instance for BONMIN 1 and 2 instances for BONMIN 50.

TABLE 3
Results for Hydro Unit Commitment and Scheduling problem.

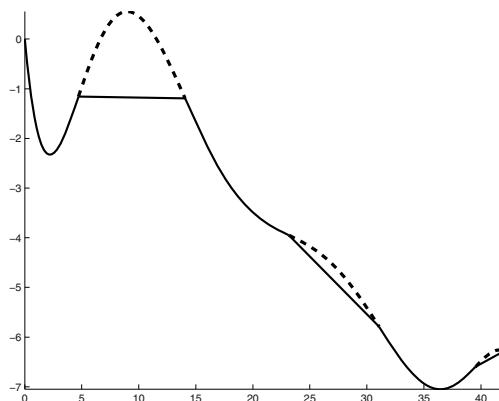
instance	var/int/cons	#	iter	LB	UB	int change	time MINLP	# br added
hydro_1	324/142/445	1	-10,231.039	-10,140.763	-		18.02	-
	332/146/449	2	-10,140.760	-10,140.763	no		23.62	4
hydro_2	324/142/445	1	-3,950.697	-3,891.224	-		21.73	-
	...	2	-3,950.583	-3,891.224	no		21.34	2
	...	3	-3,950.583	-3,891.224	no		27.86	2
hydro_3	336/148/451	4	-3,932.182	-3,932.182	no		38.20	2
	324/142/445	1	-4,753.849	-4,634.409	-		59.33	-
	...	2	-4,719.927	-4,660.189	no		96.93	4
	336/148/451	3	-4,710.734	-4,710.734	yes		101.57	2

TABLE 4
Results for Hydro Unit Commitment and Scheduling problem.

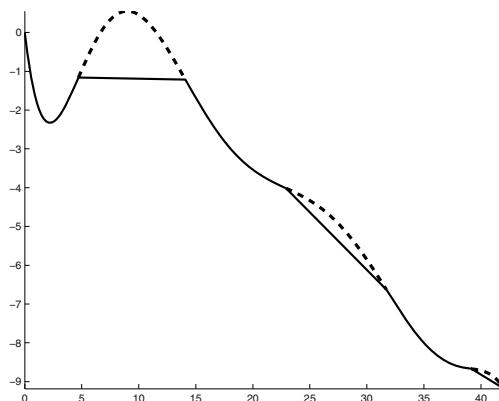
instance	var/int/cons original	SC-MINLP		COUENNE		BONMIN 1		BONMIN 50	
		time (LB)	UB	time (LB)	UB	time	UB	time	UB
hydro_1	124/62/165	107.77	-10,140.763	(-11,229.80)	-10,140.763	5.03	-10,140.763	5.75	-7,620.435
hydro_2	124/62/165	211.79	-3,932.182	(-12,104.40)	-2,910.910	4.63	-3,928.139	7.02	-3,201.780
hydro_3	124/62/165	337.77	-4,710.734	(-12,104.40)	-3,703.070	5.12	-4,131.095	13.76	-3,951.199



(a) Instance hydro_1



(b) Instance hydro_2



(c) Instance hydro_3

FIG. 6. Hydro UC: Shapes of $-\varphi(q_{jt})$ for the three instances.

TABLE 5
Results for Nonlinear Continuous Knapsack problem.

instance	var/int/cons	iter #	LB	UB	int change	time MINLP	# br added
nck_20_100	144/32/205	1	-162.444	-159.444	-	0.49	-
	146/33/206	2	-159.444	-159.444	-	0.94	1
nck_20_200	144/32/205	1	-244.015	-238.053	-	0.67	-
	...	2	-241.805	-238.053	-	0.83	1
	...	3	-241.348	-238.053	-	1.16	1
	...	4	-240.518	-238.053	-	1.35	1
	...	5	-239.865	-238.053	-	1.56	1
	...	6	-239.744	-238.053	-	1.68	1
	156/38/211	7	-239.125	-239.125	-	1.81	1
nck_20_450	144/32/205	1	-391.499	-391.337	-	0.79	-
	146/32/206	2	-391.364	-391.337	-	0.87	1
nck_50_400	356/78/507	1	-518.121	-516.947	-	4.51	-
	...	2	-518.057	-516.947	-	14.94	2
	...	3	-517.837	-516.947	-	23.75	2
	...	4	-517.054	-516.947	-	25.07	2
	372/86/515	5	-516.947	-516.947	-	31.73	2
nck_100_35	734/167/1035	1	-83.580	-79.060	-	3.72	-
	...	2	-82.126	-81.638	-	21.70	2
	...	3	-82.077	-81.638	-	6.45	2
	744/172/1040	4	-81.638	-81.638	-	11.19	1
nck_100_80	734/167/1035	1	-174.841	-171.024	-	6.25	-
	...	2	-173.586	-172.631	-	24.71	2
	742/171/1039	3	-172.632	-172.632	-	12.85	2

TABLE 6
Results for Nonlinear Continuous Knapsack problem.

instance	var/int/cons original	SC-MINLP		COUENNE		BONMIN 1		BONMIN 50	
		time (LB)	UB	time (LB)	UB	time	UB	time	UB
nck_20_100	40/0/21	15.76	-159.444	3.29	-159.444	0.02	-159.444	1.10	-159.444
nck_20_200	40/0/21	23.76	-239.125	(-352.86)	-238.053	0.03	-238.053	0.97	-239.125
nck_20_450	40/0/21	15.52	-391.337	(-474.606)	-383.149	0.07	-348.460	0.84	-385.546
nck_50_400	100/0/51	134.25	-516.947	(-1020.73)	-497.665	0.08	-438.664	2.49	-512.442
nck_100_35	200/0/101	110.25	-81.638	90.32	-81.638	0.04	-79.060	16.37	-79.060
nck_100_80	200/0/101	109.22	-172.632	(-450.779)	-172.632	0.04	-159.462	15.97	-171.024

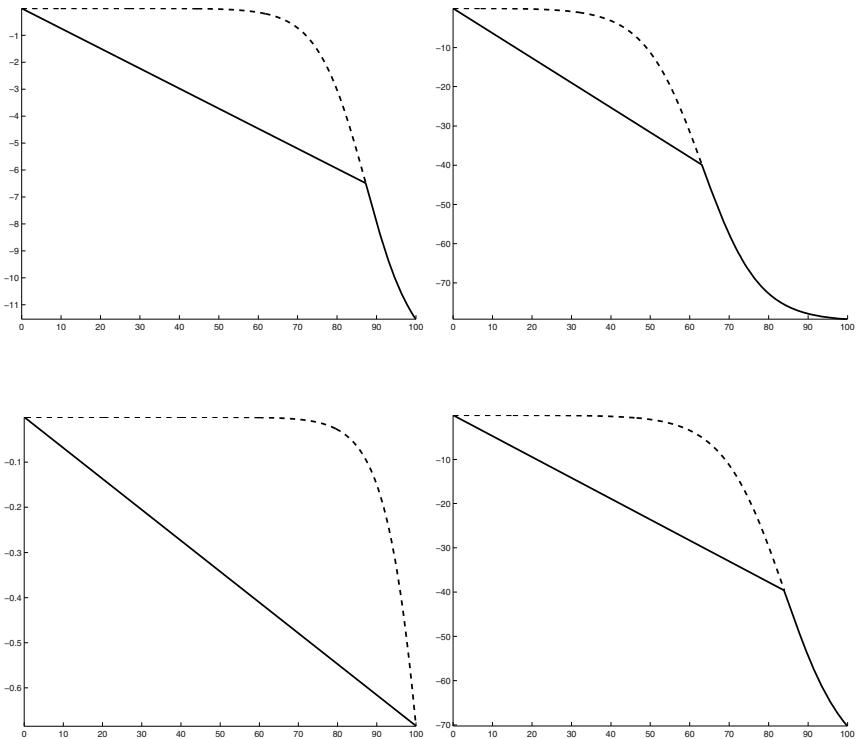


FIG. 7. Example shapes of $-g_j(x_j)$ for instance nck_20_100.

3.4. GLOBALLib and MINLPLib instances. Our algorithm is suitable for problems with functions having all non-convexities manifested as sums of univariate non-convex functions, and moreover the variables in the univariate non-convex functions should be bounded. We selected 16 instances from GLOBALLib [11] and 9 from MINLPLib [15] to test our algorithm. These instances were selected because they were easily reformulated to have these properties. Note that we are not advocating solution of general global optimization problems via reformulation as problems of the form \mathcal{P} , followed by application of our algorithm SC-MINLP. We simply produced these reformulations to enlarge our set of test problems.

To reformulate these instances in order to have univariate non-convex functions, we used as a starting point the reformulation performed by COUENNE [2]. It reformulates MINLP problems in order to have the following “basic operators”: sum, product, quotient, power, exp, log, sin, cos, abs. The only non-univariate basic operators are product and quotient (power x^y is converted to $e^{y \log(x)}$). In these cases we modified the reformulation of COUENNE in order to have only univariate non-convexities:

- product: xy is transformed into $(w_1^2 - w_2^2)/4$ with $w_1 = x + y$ and $w_2 = x - y$;
- quotient: x/y is transformed into xw and $w = 1/y$ and xw is then treated as any other product.

We report in the first part of the table the GLOBALLib results. The second part of the table reports the MINLPLib results. The instances were selected among the ones reported in the computational results of a recent paper [2]. Some of these particular MINLPLib instances are originally convex, but their reformulation is non-convex. It is clear that the employment of specific solvers for convex MINLPs such as BONMIN to solve this kind of problem is much more efficient and advisable. However, we used the non-convex reformulation of these instances to extend our test bed. In [Table 8](#), the second column reports the number of variables, integer variables and constraints of the reformulated problem.

[Table 8](#) shows that COUENNE performs much better than SC-MINLP on the GLOBALLib instances that we tested. But on such small instances, COUENNE behaves very well in general, so there cannot be much to recommend any alternative algorithm. Also BONMIN 1 and BONMIN 50 perform well. They find global optima in 10 (resp. 13) out of the 16 GLOBALLib instances rather quickly. However, in 3 (resp. 1) instances they fail to produce a feasible solution.

Concerning the MINLPLib instances, 4 out of 9 instances are solved to global optimality by SC-MINLP. COUENNE finishes within the imposed time limit on precisely the same 4 instances (in one case, COUENNE with default settings incorrectly returned a solution with a worse objective function). In the 5 instances not solved (by both solvers) within the imposed time limit, the lower bound given by SC-MINLP is always better (higher) than the one provided by COUENNE. This result emphasizes the quality of the lower bound computed by the solution of the convex MINLP relaxation \mathcal{Q} . Moreover, the upper bound computed by SC-MINLP is better in 2 instances out of 5. Concerning BONMIN 1 and BONMIN 50 performance, when they terminate before the time limit is reached (4 instances out of 9), the solution computed is globally optimal for 3 instances. Note that in these cases, the CPU time needed by BONMIN 1 and BONMIN 50 is often greater than that needed by SC-MINLP. When the time limit is reached, BONMIN 1 computes a better solution than SC-MINLP in only 1 instance out of 5 (for 1 instance they provide the same solution) and BONMIN 50 computes a better solution than SC-MINLP in 1 only instance out of 5.

Finally, we note that for several instances of applying SC-MINLP, in just the second iteration, the convex MINLP solver BONMIN was in the midst of working when our self-imposed time limit of 2 hours was reached. In many of these cases, much better lower bounds could be achieved by increasing the time limit. Generally, substantial improvements in BONMIN would produce a corresponding improvement in the results obtained with SC-MINLP.

TABLE 7
Results for GLOBAALLib and MINNPLib.

reformulated instance	var/int/cons	iter #	LB	UB	int change	time MINLP	# br added
ex14_2_1	239/39/358	1	0.000	0.000	-	5.35	-
ex14_2_2	110/18/165	1	0.000	0.000	-	3.33	-
ex14_2_6	323/53/428	1	0.000	0.000	-	7.02	-
ex14_2_7	541/88/808	1	0.000	0.000	-	1.06	-
ex2_1_1	27/5/38	1	-18.900	-16.500	-	0.01	-
	...	2	-18.318	-16.500	-	0.06	1
	...	3	-18.214	-16.500	-	0.12	1
	...	4	-18.000	-16.500	-	0.15	1
	...	5	-17.625	-17.000	-	0.22	2
	39/11/44	6	-17.000	-17.000	-	0.26	1
ex2_1_2	29/5/40	1	-213.000	-213.000	-	0.01	-
ex2_1_3	36/4/41	1	-15.000	-15.000	-	0.00	-
ex2_1_4	15/1/16	1	-11.000	-11.000	-	0.00	-
ex2_1_5	50/7/72	1	-269.453	-268.015	-	0.01	-
	54/9/74	2	-268.015	-268.015	-	0.15	2
ex2_1_6	56/10/81	1	-44.400	-29.400	-	0.01	-
	...	2	-40.500	-39.000	-	0.16	2
	...	3	-40.158	-39.000	-	0.25	1
	66/15/86	4	-39.000	-39.000	-	0.52	2

TABLE 7 (*Continued*)

ex2_1_7	131/20/181	1	-13,698,362	-4,105,278	-	0.07	-
		2	-10,643,558	-4,105,278	-	1.34	11
		3	-8,219,738	-4,105,278	-	2.68	5
		4	-6,750,114	-4,105,278	-	4.66	10
		5	-5,450,142	-4,105,278	-	16.42	11
		6	-5,014,019	-4,105,278	-	32.40	6
		7	-4,740,743	-4,105,278	-	38.61	15
		8	-4,339,192	-4,150,410	-	86.47	4
		9	-4,309,425	-4,150,410	-	133.76	11
		10	-4,250,248	-4,150,410	-	240.50	6
		11	-4,156,125	-4,150,410	-	333.08	5
		12	-4,150,411	-4,150,410	-	476.47	5
ex9_2_2	309/109/270	1	55,556	100,000	-	0.00	-
	68/14/97	2	78,679	100,000	-	0.83	10
	...	3	95,063	100,000	-	5.77	19
	...	4	98,742	100,000	-	18.51	8
	...	5	99,684	100,000	-	37.14	11
	184/72/155	6	99,960	100,000	-	56.00	10
ex9_2_3	90/18/125	1	-30,000	0,000	-	0.00	-
	...	2	-30,000	0,000	-	2.29	12
	...	3	-30,000	0,000	-	6.30	12
	...	4	-30,000	0,000	-	8.48	12
	...	5	-24,318	0,000	-	30.44	22

TABLE 7 (*Continued*)

	...	6	-23.393	0.000	-	16.25	10
	...	7	-21.831	0.000	-	31.92	12
	...	8	-19.282	0.000	-	26.53	12
	...	9	-7.724	0.000	-	203.14	17
	332/139/246	10	-0.000	0.000	-	5,426.48	12
ex9_2_6	105/22/149	1	-1.500	3.000	-	0.02	-
	...	2	-1.500	1.000	-	9.30	28
	...	3	-1.500	0.544	-	34.29	15
	...	4	-1.500	-1.000	-	48.33	12
	...	5	-1.500	-1.000	-	130.04	22
	...	6	-1.500	-1.000	-	50.29	24
	...	7	-1.500	-1.000	-	53.66	13
	...	8	-1.500	-1.000	-	67.93	22
	...	9	-1.500	-1.000	-	103.13	14
	...	10	-1.366	-1.000	-	127.60	12
	...	11	-1.253	-1.000	-	1,106.78	22
	...	12	-1.116	-1.000	-	3,577.48	13
	...	13	-1.003	-1.000	-	587.11	15
	557/248/375	14	-1.003	-1.000	-	1,181.12	14
ex5_2_5	802/120/1,149	1	-34,200.833	-3,500.000	-	0.10	-
	1,280/359/1,388	2	-23,563.500	-3,500.000	-	7,192.22	239
ex5_3_3	1,025/181/1,474	1	-67,499.021	-	-	0.17	-
	1,333/335/1,628	2	-16,872.900	3.056	-	7,187.11	154

TABLE 7 (*Continued*)

du-opt	458/18/546	1	3.556	3.556	-	5.30	-
du-opt5	455/15/545	1	8.073	8.073	-	9.38	-
f07	366/42/268	1	8.759	22.518	-	7,200	-
m6	278/30/206	1	82.256	82.256	-	182.72	-
no7_ar2_1	421/41/325	1	90.583	127.774	-	7,200	-
no7_ar3_1	421/41/325	1	81.539	107.869	-	7,200	-
no7_ar4_1	421/41/325	1	76.402	104.534	-	7,200	-
o7_2	366/42/268	1	79.365	124.324	-	7,200	-
stockcycle	626/432/290	1	119,948.675	119,948.676	-	244.67	-

TABLE 8
Results for GLOBALLib and MINPLib.

reformulated instance	var/int/cons original	SC-MINLP		COUENNE		BONMIN 1		BONMIN 50	
		time (LB)	UB	time (LB)	UB	time	UB	time	UB
ex14.2_1	122/0/124	21.81	0.000	0.18	0.000	0.13	0.000	46.25	0.000
ex14.2_2	56/0/57	12.86	0.000	0.10	0.000	0.05	0.000	28.02	0.000
ex14.2_6	164/0/166	42.56	0.000	1.03	0.000	1.25	0.000	138.02	0.000
ex14.2_7	277/0/280	128.70	0.000	0.63	0.000	0.28	0.000	170.62	0.000
ex2.1_1	12/0/8	2.84	-17.000	0.16	-17.000	0.04	-11.925	0.89	-17.000
ex2.1_2	14/0/10	1.75	-213.000	0.06	-213.000	0.02	-213.000	0.66	-213.000
ex2.1_3	24/0/17	1.62	-15.000	0.04	-15.000	0.02	-15.000	0.36	-15.000
ex2.1_4	12/0/10	1.28	-11.000	0.04	-11.000	0.03	-11.000	0.58	-11.000
ex2.1_5	29/0/30	2.21	-268.015	0.13	-268.015	0.03	-268.015	0.96	-268.015
ex2.1_6	26/0/21	3.31	-39.000	0.12	-39.000	0.04	-39.000	1.11	-39.000
ex2.1_7	71/0/61	1388.00	4,150.410	3.98	4,150.410	0.04	4,150.410	13.64	4,150.410
ex9.2_2	38/0/37	1,355.47	100.000	0.36	100.000	0.12	-	2.65	100.000
ex9.2_3	54/0/53	5,755.76	0.000	0.73	0.000	0.08	15.000	2.85	5.000
ex9.2_6	57/0/53	(-1.003)	-1.000	0.78	-1.000	0.21	-	4.04	-1.000
ex5.2_5	442/0/429	(-23.563, 500)	-3,500.000	(-11.975, 600)	-3,500.000	2.06	-3,500.000	315.53	-3,500.000
ex5.3_3	563/0/550	(-16.872, 900)	3.056	(-16,895, 400)	3.056	20.30	-	22.92	-
du-opt	242/18/230	13.52	3.556	38.04	3.556	41.89	3.556	289.88	3.556
du-opt5	239/15/227	17.56	8.073	37.96	8.073	72.32	8.118	350.06	8.115
f67	338/42/437	(8.759)	22.518	(1.95)	22.833	-	22.518	-	24.380
m6	254/30/327	185.13	82.256	54.13	82.256	154.42	82.256	211.49	82.256
no7_ar2_1	394/41/551	(90.583)	127.774	(73.78)	111.141	-	122.313	-	107.871
no7_ar3_1	394/41/551	(81.539)	107.869	(42.19)	113.810	-	121.955	-	119.092
no7_ar4_1	394/41/551	(76.402)	104.534	(54.85)	98.518	-	117.992	-	124.217
o7_2	338/42/437	(79.365)	124.324	(5.85)	133.988	-	126.674	-	130.241
stockcycle	578/480/195	251.54	119,948.676	84.35	119,948.676*	321.89	119,948.676	328.03	119,948.676

* This time and correct solution were obtained with non-default options of Couenne (which failed with default settings).

4. Summary. In this paper, we proposed an algorithm for solving to global optimality a broad class of separable MINLPs. Our simple algorithm, implemented within the AMPL modeling language, works with a lower-bounding convex MINLP relaxation and an upper-bounding non-convex NLP restriction. For the definition of the lower-bounding problem, we identify subintervals of convexity and concavity for the univariate functions using external calls to MATLAB; then we develop a convex MINLP relaxation of the problem approximating the concave intervals of each non-convex function with the linear relaxation. The subintervals are glued together using binary variables. We iteratively refine our convex MINLP relaxation by modifying it at the modeling level. The upper-bound is obtained by fixing the integer variables in the original non-convex MINLP, then locally solving the associated non-convex NLP restriction. We presented preliminary computational experiments on models from a variety of application areas, including problems from GLOBALLib and MINLPLib. We compared our algorithm with the open-source solvers COUENNE as an exact approach, and BONMIN as a heuristic approach, obtaining significant success.

Acknowledgments. This work was partially developed when the first author was visiting the IBM T.J. Watson Research Center, and their support is gratefully acknowledged. We also thank Pietro Belotti for discussions about the modification of COUENNE to reformulate GLOBALLib and MINLPLib instances.

REFERENCES

- [1] E. BEALE AND J. TOMLIN, *Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables*, in Proc. of the 5th Int. Conf. on Operations Research, J. Lawrence, ed., 1970, pp. 447–454.
- [2] P. BELOTTI, J. LEE, L. LIBERTI, F. MARGOT, AND A. WÄCHTER, *Branching and bounds tightening techniques for non-convex MINLP*, Optimization Methods and Software, **24** (2009), pp. 597–634.
- [3] P. BONAMI, L. BIEGLER, A. CONN, G. CORNUÉJOLS, I. GROSSMANN, C. LAIRD, J. LEE, A. LODI, F. MARGOT, N. SAWAYA, AND A. WÄCHTER, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization, **5** (2008), pp. 186–204.
- [4] BONMIN. projects.coin-or.org/Bonmin, v. 1.0.
- [5] A. BORGHETTI, C. D'AMBROSIO, A. LODI, AND S. MARTELLO, *An MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir*, IEEE Transactions on Power Systems, **23** (2008), pp. 1115–1124.
- [6] COUENNE. projects.coin-or.org/Couenne, v. 0.1.
- [7] C. D'AMBROSIO, J. LEE, AND A. WÄCHTER, *A global-optimization algorithm for mixed-integer nonlinear programs having separable non-convexity*, in Proc. of 17th Annual European Symposium on Algorithms (ESA), Copenhagen, Denmark. *Lecture Notes in Computer Science*, A. Fiat and P. Sander, eds., **5757** (2009), pp. 107–118.
- [8] M. DURAN AND I. GROSSMANN, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Mathematical Programming, **36** (1986), pp. 307–339.

- [9] R. FLETCHER AND S. LEYFFER, *Solving mixed integer nonlinear programs by outer approximation*, Mathematical Programming, **66** (1994), pp. 327–349.
- [10] R. FOURER, D. GAY, AND B. KERNIGHAN, *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press/Brooks/Cole Publishing Co., second ed., 2003.
- [11] GLOBALLIB. www.gamsworld.org/global/globallib.htm.
- [12] O. GÜNLÜK, J. LEE, AND R. WEISMANTEL, *MINLP strengthening for separable convex quadratic transportation-cost UFL*, 2007. IBM Research Report RC24213.
- [13] L. LIBERTI, *Writing global optimization software*, in Global Optimization: From Theory to Implementation, L. Liberti and N. Maculan, eds., Springer, Berlin, 2006, pp. 211–262.
- [14] MATLAB. www.mathworks.com/products/matlab/, R2007a.
- [15] MINPLIB. www.gamsworld.org/minlp/minplib.htm.
- [16] I. NOWAK, H. ALPERIN, AND S. VIGERSKE, *LaGO – an object oriented library for solving MINLPs*, in Global Optimization and Constraint Satisfaction, vol. 2861 of Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 2003, pp. 32–42.
- [17] I. QUESADA AND I. GROSSMANN, *An LP/NLP based branch and bound algorithm for convex MINLP optimization problems*, Computer & Chemical Engineering, **16** (1992), pp. 937–947.
- [18] N. SAHINIDIS, *BARON: A general purpose global optimization software package*, J. Global Opt., **8** (1996), pp. 201–205.
- [19] A. WÄCHTER AND L.T. BIEGLER, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Mathematical Programming, **106** (2006), pp. 25–57.

GLOBAL OPTIMIZATION OF MIXED-INTEGER SIGNOMIAL PROGRAMMING PROBLEMS

ANDREAS LUNDELL* AND TAPIO WESTERLUND*

Abstract. Described in this chapter, is a global optimization algorithm for mixed-integer nonlinear programming problems containing signomial functions. The method obtains a convex relaxation of the nonconvex problem through reformulations using single-variable transformations in combination with piecewise linear approximations of the inverse transformations. The solution of the relaxed problems converges to the global optimal solution as the piecewise linear approximations are improved iteratively. To illustrate how the algorithm can be used to solve problems to global optimality, a numerical example is also included.

Key words. Nonconvex optimization, mixed-integer nonlinear programming, signomial functions, convex reformulations.

AMS(MOS) subject classifications. 90C11, 90C26, 90C30.

1. Introduction. In this chapter, an algorithm for finding the global optimal solution to mixed-integer nonlinear programming (MINLP) problems containing signomial functions is described. The algorithm employs single-variable transformations convexifying the nonconvex signomial functions termwise. When the inverse transformations are approximated with piecewise linear functions, the feasible region of the reformulated problem is overestimated, *i.e.*, the solution of this problem is a lower bound of the solution of the original problem. Also included in the technique for solving these kinds of problems to global optimality is a preprocessor for determining an optimized set of transformations. By adding new breakpoints to the piecewise linear functions, the feasible region can be made tighter and tighter until the global optimal solution of the original problem is obtained.

Optimization problems involving signomial functions appear in many different application areas, since, for instance, all polynomial, bilinear and trilinear functions are special cases of signomials. Some special applications are in design of heat exchanger networks [5, 7], chemical reactors [6], optimal condensers [2], delta-sigma modulator topologies [11] and inductors [12], as well as, chemical equilibrium [24], optimal control [13], image restoration [28] and trim-loss minimization [10] problems.

The results presented in this chapter is a review over previously found results from, *e.g.*, [15, 16, 18, 25, 31, 33].

2. The problem formulation. The class of MINLP problems which can be solved to global optimality using the techniques in this chapter is of the mixed-integer signomial programming (MISP) type.

*Process Design and Systems Engineering, Åbo Akademi University, FIN-20500 Turku, Finland ({andreas.lundell, tapio.westerlund}@abo.fi).

DEFINITION 2.1. A MISP problem can be formulated as

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}), \\ & \text{subject to} && \mathbf{Ax} = \mathbf{a}, \quad \mathbf{Bx} \leq \mathbf{b}, \\ & && \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \\ & && \mathbf{q}(\mathbf{x}) + \boldsymbol{\sigma}(\mathbf{x}) \leq \mathbf{0}. \end{aligned} \tag{2.1}$$

The vector of variables $\mathbf{x} = [x_1, x_2, \dots, x_N]$ may consist of both continuous and discrete variables. The objective function f is assumed to be convex and $\mathbf{Ax} = \mathbf{a}$ and $\mathbf{Bx} \leq \mathbf{b}$ are linear equality and inequality constraints respectively. The constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ are nonlinear convex inequality constraints, and the constraints $\mathbf{q}(\mathbf{x}) + \boldsymbol{\sigma}(\mathbf{x}) \leq \mathbf{0}$ are generalized signomial constraints, consisting of sums of convex functions $\mathbf{q}(\mathbf{x})$ and signomial functions $\boldsymbol{\sigma}(\mathbf{x})$. A signomial objective function $s(\mathbf{x})$ can also be used, by introducing the variable μ as the new objective function, and including the original objective function as the signomial constraint $s(\mathbf{x}) - \mu \leq 0$.

A signomial function is a sum of signomial terms, where each term consists of products of power functions. Thus, a signomial function of N variables and J signomial terms can be expressed mathematically as

$$\boldsymbol{\sigma}(\mathbf{x}) = \sum_{j=1}^J c_j \prod_{i=1}^N x_i^{p_{ji}}, \tag{2.2}$$

where the coefficients c_j and the powers p_{ji} are real-valued. A posynomial term is a positive signomial term, so signomial functions are generalizations of posynomial functions, since the terms are allowed to be both positive and negative. Note that, if a certain variable x_i does not exist in the j -th term, then $p_{ji} = 0$.

The variables x_i are allowed to be reals or integers. Since imaginary solutions to the problems are not allowed, negative values on a variable appearing in a signomial term with noninteger power must be excluded. Also, zero has to be excluded in case a power is negative. Thus, all variables occurring in the signomial terms are assumed to have a positive fixed lower bound. For variables having a lower bound of zero, this bound may be approximated with a small positive lower bound of $\epsilon > 0$. Furthermore, translations of the form $x'_i = x_i + \tau_i$, where $\tau_i > |\min x_i|$, may be used for variables with a nonpositive lower bound. Note however, that using translations may introduce additional variables and signomial terms into the problem.

Signomials are often highly nonlinear and nonconvex. Unfortunately, convex envelopes are only known for some special cases, for instance, so-called McCormick envelopes for bilinear terms [22]. Therefore, other techniques for dealing with optimization problems containing signomial functions are needed, including the α BB underestimator [1, 8] or the methods

used in BARON [27, 29]. These techniques are not confined to convexifying signomial functions only, rather they can be applied to larger classes of nonconvex functions.

In the transformation method presented here, based on single-variable power and exponential transformations, the convexity of signomial functions is guaranteed termwise. This is, however, only a sufficient condition for convexity; for instance, the signomial function $f(x_1, x_2) = x_1^2 + 2x_1x_2 + x_2^2$ is convex although the middle term is nonconvex. In the next theorem convexity conditions for signomial terms, first derived in [21], are given.

THEOREM 2.1. *The positive signomial term $s(\mathbf{x}) = c \cdot x_1^{p_1} \cdots x_N^{p_N}$, where $c > 0$, is convex if one of the following two conditions is fulfilled: (i) all powers p_i are negative, or (ii) one power p_k is positive, the rest of the powers p_i , $i \neq k$ are negative and the sum of the powers is greater than or equal to one, i.e.,*

$$\sum_{i=1}^N p_i \geq 1. \quad (2.3)$$

The negative signomial term $s(\mathbf{x}) = c \cdot x_1^{p_1} \cdots x_N^{p_N}$, where $c < 0$, is convex if all powers p_i are positive and the sum of the powers is between zero and one, i.e.,

$$0 \leq \sum_{i=1}^N p_i \leq 1. \quad (2.4)$$

By using the previous theorem, it is easy to determine the convexity of any signomial term, which is illustrated in the following example.

EXAMPLE 1. *For $x_1, x_2 > 0$, the following signomial terms are convex*

$$\frac{1}{x_1x_2} = x_1^{-1}x_2^{-1}, \quad \frac{x_1^2}{x_2} = x_1^2x_2^{-1} \quad \text{and} \quad -\sqrt{x_1} = -x_1^{0.5} \quad (2.5)$$

while the following terms are nonconvex

$$\frac{x_1}{x_2} = x_1x_2^{-1}, \quad \sqrt{x_1} = x_1^{0.5} \quad \text{and} \quad -x_1x_2. \quad (2.6)$$

In Section 3, it is explained how it is possible to deduce single-variable power transformations to convexify general nonconvex signomial terms using Theorem 2.1.

2.1. Piecewise linear functions using special ordered sets. In the transformation technique presented in the next section, piecewise linear functions (PLFs) play an important role. There are many different ways of expressing PLFs; here, one using so-called special ordered sets (SOS) is

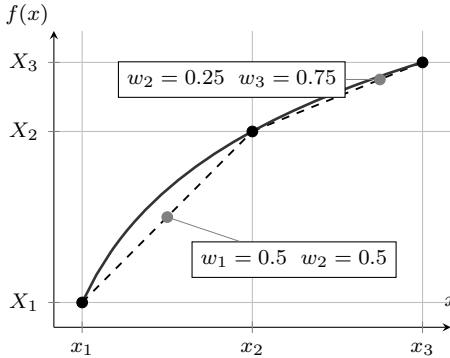


Fig. 1: A PLF using the SOS type 2 formulation.

given, but other variants, including methods using binary variables, can be found in [9]. However, SOS formulations are often computationally more efficient in optimization problems than those using binary variables.

A *special ordered set of type 2* is defined as a set of integers, continuous or mixed-integer and continuous variables, where at most two variables in the set are nonzero, and if there are two nonzero variables, these must be adjacent in the set. For example, the sets $\{1, 0, \dots, 0\}$ and $\{0, a, b, 0, \dots, 0\}$, $a, b \in R$ are SOS type 2 sets.

In the PLF formulation in [3] presented here, one SOS of type 2, $\{w_k\}_{k=1}^K$, is used, with the additional conditions that all variables w_k assume positive real values between zero and one, and that the sum of the variables in the set is equal to one, *i.e.*,

$$\forall k = 1, \dots, K : \quad 0 \leq w_k \leq 1 \quad \text{and} \quad \sum_{k=1}^K w_k = 1. \quad (2.7)$$

The variable $x \in [\underline{x}, \bar{x}]$ can then be expressed as

$$x = \sum_{k=1}^K x_k w_k, \quad (2.8)$$

and the PLF approximating the function f in the interval $[\underline{x}, \bar{x}]$ becomes

$$\hat{f}(x) = \sum_{k=1}^K X_k w_k, \quad (2.9)$$

where the function f assumes the values $X_k = f(x_k)$ at the K consecutive points $x_k \in [\underline{x}, \bar{x}]$, $x_1 < x_2 < \dots < x_K$. In this formulation, only one

additional variable w_k is required for each breakpoint added to the PLF. A PLF approximation using the SOS type 2 formulation is illustrated in Figure 1.

3. The transformation technique. In this section, the transformation technique for signomial functions in MINLP problems of the type in Definition 2.1 is presented in more detail. First, the transformation schemes for positive and negative signomial terms are presented, and then some theoretical results regarding the relationships between the different transformations are given.

The transformation procedure for an individual generalized signomial constraint can be illustrated as follows

$$\begin{aligned} q_m(\mathbf{x}) + \sigma_m(\mathbf{x}) &\leq 0 \xrightarrow{(i)} q_m(\mathbf{x}) + \sigma_m^C(\mathbf{x}, \mathbf{X}) \\ &\leq 0 \xrightarrow{(ii)} q_m(\mathbf{x}) + \sigma_m^C(\mathbf{x}, \hat{\mathbf{X}}) \leq 0. \end{aligned} \quad (3.1)$$

In step (i), the nonconvex signomial terms in the signomial function $q_m(\mathbf{x})$ are convexified using single-variable transformations $x_i = T_{ji}(X_{ji})$, where X_{ji} is the new transformation variable. In this step, the problem is reformulated so that the generalized signomial constraints are convex, but the problem itself is still nonconvex, since the nonlinear equality constraints representing the relations between the transformation variables and the original variables, *i.e.*, $X_{ji} = T_{ji}^{-1}(x_i)$, must be included. In step (ii), however, these relations are approximated with PLFs, in such a way that the relaxed and convex feasible region of the transformed problem will overestimate that of the original problem. Thus, the solution to the transformed problem will be a lower bound of the original problem. The lower bound can then be improved by iteratively adding more breakpoints to the PLFs and, if the breakpoints are chosen in a certain way, the solution of the approximated problems will form a converging sequence to the global optimal solution of the original nonconvex problem.

The transformation technique using single-variable transformations have been studied previously in many papers, *e.g.*, [4, 15, 25, 31].

3.1. Transformations for positive terms. A positive signomial term is convexified using single-variable power transformations (PTs), as well as underestimated by expressing the relation between the original and transformation variables with PLFs according to

$$s(\mathbf{x}) = c \prod_i x_i^{p_i} = c \prod_{i:p_i < 0} x_i^{p_i} \cdot \prod_{i:p_i > 0} X_i^{p_i Q_i} = s_C(\mathbf{x}, \mathbf{X}) \geq s_C(\mathbf{x}, \hat{\mathbf{X}}), \quad (3.2)$$

as long as the transformation powers Q_i fulfill certain conditions. There are two different types of transformation schemes using PTs – the negative power transformation (NPT) and positive power transformation (PPT) – corresponding to the two different cases for the convexity of positive terms

in Theorem 2.1. Note that, for a positive term, only the variables x_i having a negative power p_i must be transformed.

DEFINITION 3.1. *The NPT convex underestimator for a positive signomial term is obtained by applying the transformation*

$$x_i = X_i^{Q_i}, \quad Q_i < 0, \quad (3.3)$$

to all variables x_i with positive powers ($p_i > 0$) as long as the inverse transformation $X_i = x_i^{1/Q_i}$ is approximated by a PLF \hat{X}_i .

The transformation technique in [14] and [30] is similar to the NPT, in the respect that it employs single-variable PTs with negative powers approximated by linear functions; it is, however designed to be used in a branch-and-bound type framework.

DEFINITION 3.2. *The PPT convex underestimator for a positive signomial term is obtained by applying the transformation*

$$x_i = X_i^{Q_i}, \quad (3.4)$$

to all variables with positive powers, where the transformation powers $Q_i < 0$ for all indices i , except for one ($i = k$), where $Q_k \geq 1$. Furthermore, the condition

$$\sum_{i:p_i>0} p_i Q_i + \sum_{i:p_i<0} p_i \geq 1 \quad (3.5)$$

must be fulfilled and the inverse transformation $X_i = x_i^{1/Q_i}$ approximated by a PLF \hat{X}_i .

There is also another transformation scheme available for convexifying positive signomial terms, namely the exponential transformation (ET). The single-variable ET has been used for a long time for reformulation of nonconvex geometric programming problems. This transformation is based on the fact that the function

$$f(\mathbf{x}) = c \cdot e^{p_1 x_1 + p_2 x_2 + \dots + p_i x_i} \cdot x_{i+1}^{p_{i+1}} x_{i+2}^{p_{i+2}} \cdots x_I^{p_I}, \quad (3.6)$$

where $c > 0$, $p_1, \dots, p_i > 0$ and $p_{i+1}, \dots, p_I < 0$, is convex on \mathbb{R}_+^n . Using the ET, a nonconvex positive signomial term is convexified and underestimated according to

$$s(\mathbf{x}) = c \prod_i x_i^{p_i} = c \prod_{i:p_i<0} x_i^{p_i} \cdot \prod_{i:p_i>0} e^{p_i X_i} = s_C(\mathbf{x}, \mathbf{X}) \geq s_C(\mathbf{x}, \hat{\mathbf{X}}). \quad (3.7)$$

As in the NPT and PPT, only the variables x_i having a positive power p_i must be transformed.

DEFINITION 3.3. *The ET convex underestimator for a positive signomial term is obtained by applying the transformation*

$$x_i = e^{X_i} \quad (3.8)$$

to the individual variables with positive powers as long as the inverse transformation $X_i = \ln x_i$ is approximated by a PLF \hat{X}_i .

3.1.1. Examples of the transformation technique. In the following two examples, the transformations in Definitions 3.1–3.3 are used to find convex underestimators for two signomial functions.

EXAMPLE 2. *The convex underestimator for the function $f(x_1, x_2) = x_1 x_2$ obtained when applying the ET to the function is given by*

$$\hat{f}_E(\hat{X}_{1,E}, \hat{X}_{2,E}) = e^{\hat{X}_{1,E}} e^{\hat{X}_{2,E}}, \quad (3.9)$$

where the inverse transformations $X_{1,E} = \ln x_1$ and $X_{2,E} = \ln x_2$ have been replaced with the PLF approximations $\hat{X}_{1,E}$ and $\hat{X}_{2,E}$ respectively. Applying either of the PTs to the function, gives the convex underestimator

$$\hat{f}_P(\hat{X}_{1,P}, \hat{X}_{2,P}) = \hat{X}_{1,P}^{Q_1} \hat{X}_{2,P}^{Q_2}, \quad (3.10)$$

where the inverse transformations $X_{1,P} = x_1^{1/Q_1}$ and $X_{2,P} = x_2^{1/Q_2}$ have been replaced with the PLF approximations $\hat{X}_{1,P}$ and $\hat{X}_{2,P}$ respectively. If the transformation used is the NPT, both Q_1 and Q_2 must be negative, and if the PPT is used, one of these must be positive, the other negative and $Q_1 + Q_2 \geq 1$. For example, $Q_1 = Q_2 = -1$ or $Q_1 = 2$ and $Q_2 = -1$ gives convex underestimators for $f(x_1, x_2)$.

EXAMPLE 3. *The convex underestimator for the function $f(x_1, x_2) = x_1/x_2 = x_1 x_2^{-1}$ obtained when applying the ET to the function is given by*

$$\hat{f}_E(\hat{X}_{1,E}, x_2) = e^{\hat{X}_{1,E}} x_2^{-1}, \quad (3.11)$$

where the inverse transformation $X_{1,E} = \ln x_1$ has been replaced with the PLF approximation $\hat{X}_{1,E}$, i.e., only one variable is transformed. When applying either of the PTs to the function, the convex underestimator becomes

$$\hat{f}_P(\hat{X}_{1,P}, x_2) = \hat{X}_{1,P}^{Q_1} x_2^{-1}, \quad (3.12)$$

where the inverse transformation $X_{1,P} = x_1^{1/Q_1}$ has been replaced with the PLF approximation $\hat{X}_{1,P}$. If the transformation used is the NPT, Q_1 must be negative, and if the PPT has been used, Q_1 must be positive and $Q_1 - 1 \geq 1$, i.e., $Q_1 \geq 2$. For example, $Q_1 = 2$ can be used. Also in these cases, only one transformation is needed.

3.2. Transformations for negative terms. In a similar way as for positive terms, a negative signomial term is convexified and underestimated using single-variable PTs according to

$$\begin{aligned} s(\mathbf{x}) &= c \prod_i x_i^{p_i} = c \prod_{i:p_i < 0} X_i^{p_i Q_i} \cdot \prod_{i:p_i > 0} X_i^{p_i Q_i} \\ &= s_C(\mathbf{x}, \mathbf{X}) \geq s_C(\mathbf{x}, \hat{\mathbf{X}}). \end{aligned} \quad (3.13)$$

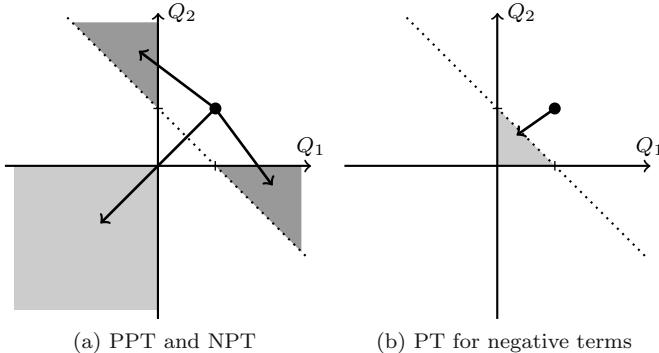


Fig. 2: Overviews of how the bilinear terms x_1x_2 and $-x_1x_2$ are transformed using the PTs into $X_1^{Q_1}X_2^{Q_2}$. The colored regions indicates convex transformations.

However, as the convexity conditions for negative terms are different according to Theorem 2.1, the requirements on the powers Q_i in the transformations are also different. For example, variables with negative powers also require transformations.

DEFINITION 3.4. *A convex underestimator for a negative signomial term is obtained by applying the transformation*

$$x_i = X_i^{Q_i}, \quad (3.14)$$

where $0 < Q_i \leq 1$ for all variables with positive powers and $Q_i < 0$ for all variables with negative power, to the individual variables in the term. Furthermore, the condition

$$0 < \sum_i p_i Q_i \leq 1, \quad (3.15)$$

must be fulfilled and the inverse transformation $X_i = x_i^{1/Q_i}$ approximated by a PLF \hat{X}_i .

3.2.1. Example of the transformation technique. In the following example, a convex underestimator for a negative signomial term is obtained using the previous definition.

EXAMPLE 4. *The convex underestimator for the function $f(x_1, x_2) = -x_1x_2^{-1}$, obtained by applying the PTs for negative terms, is given by*

$$\hat{f}_P(\hat{X}_{1,P}, \hat{X}_{2,P}) = -\hat{X}_{1,P}^{Q_1}\hat{X}_{2,P}^{-1-Q_2}, \quad (3.16)$$

where the inverse transformations $X_{1,P} = x_1^{1/Q_1}$ and $X_{2,P} = x_2^{1/Q_2}$ have been replaced with the PLF approximations $\hat{X}_{1,P}$ and $\hat{X}_{2,P}$ respectively.

In this case, Q_1 must be positive and Q_2 negative for the powers in the convexified term $1 \cdot Q_1$ and $-1 \cdot Q_2$ to both be positive. Furthermore, the sum of the powers in the transformed term needs to be less than or equal to one, so for example $Q_1 = 1/2$ and $Q_2 = -1/2$ give a valid transformation.

3.3. Relationships between the transformations. Application of the single-variable transformations together with the approximation using PLFs, results in convex underestimators of the original nonconvex term. Depending on what transformation is used, and in the case of the PTs, what the value of the transformation power Q is, different underestimation errors occur. In this section, some results regarding the relationships, connected to the underestimation properties, between the transformations are summarized. More detail on this subject, as well as proofs of the results, can be found in [15] and [18].

The first result is regarding the underestimation error resulting from transforming an individual nonconvex power function x^p .

THEOREM 3.1. *Assume that a single-variable ET and single-variable PTs with positive and negative powers, i.e., the transformations*

$$x = e^{X_E}, \quad x = X_P^{Q_P}, \quad Q_P \geq 1, \quad \text{and} \quad x = X_N^{Q_N}, \quad Q_N < 0,$$

are applied to the power function x^p , $p > 0$, where $x \in \mathbb{R}_+$ or $x \in \mathbb{Z}$ and $x \in [\underline{x}, \bar{x}]$, $\underline{x} > 0$. Then the following is true

$$\forall x \in [\underline{x}, \bar{x}] : \quad \left(\hat{X}_P^{Q_P} \right)^p \geq \left(e^{\hat{X}_E} \right)^p \geq \left(\hat{X}_N^{Q_N} \right)^p, \quad (3.17)$$

when the inverse transformations

$$X_E = \ln x, \quad X_P = x^{1/Q_P} \quad \text{and} \quad X_N = x^{1/Q_N},$$

have been replaced by the PLFs \hat{X}_E , \hat{X}_P and \hat{X}_N respectively.

Although this theorem states that any PT with positive transformation power always gives a tighter convex underestimator than the ET and the ET a tighter convex underestimator than any PT with negative transformation power, the limit of the single-variable PTs when the transformation power Q tends to plus or minus infinity is actually the single-variable ET:

THEOREM 3.2. *For the piecewise linear approximations \hat{X}_P , \hat{X}_N and \hat{X}_E of the single-variable PT with positive and negative powers and ET respectively, the following statement is true*

$$\forall x \in [\underline{x}, \bar{x}] : \quad \lim_{Q \rightarrow \infty} \hat{X}_P^Q = e^{\hat{X}_E} = \lim_{Q \rightarrow -\infty} \hat{X}_N^Q, \quad (3.18)$$

i.e., a single-variable PT with positive or negative power tends to a single-variable ET as the transformation powers tend to plus and minus infinity respectively.

Using the results from the previous theorems, the following theorem regarding the underestimation properties of a general positive signomial term can be obtained:

THEOREM 3.3. *For a general nonconvex signomial term transformed using the ET, NPT or PPT, the following statements are true:*

- (i) *The ET always gives a tighter underestimator than the NPT.*
- (ii) *The PPT gives a tighter underestimator than the NPT as long as the transformation powers $Q_{i,N}$ and $Q_{i,P}$ in the NPT and PPT respectively, fulfill the condition*

$$\forall i : p_i > 0, \quad i \neq k : \quad Q_{i,P} \leq Q_{i,N}. \quad (3.19)$$

- (iii) *Neither of the PPT nor ET gives a tighter convex underestimator in the whole domain.*

4. The SGO algorithm. The signomial global optimization (SGO) algorithm is a method for solving nonconvex MISP problems to global optimality as a sequence of overestimated convex MINLP subproblems. It is based on the generalized geometric programming extended cutting plane (GGPECP) algorithm from [31] and [33]. The SGO algorithm in the form given here was presented in [15] and [19]. One of the major differences between the original GGPECP algorithm and the newer SGO algorithm, is that the latter includes a preprocessing step, in which the single-variable transformations convexifying the signomial terms are obtained by solving a mixed-integer linear programming (MILP) problem. Furthermore, the SGO algorithm can use any convex MINLP solver for the subproblems, whereas the GGPECP algorithm, as the name indicates, uses the solver α ECP exclusively.

The SGO algorithm solves problem of the MISP type to global optimality as a sequence of converging convex subproblems. In each iteration, the approximation of the overestimated feasible region of the reformulated problem is improved by adding additional breakpoints to the PLFs describing the relationships between the transformation variables and original variables. However, as shown in [31], it is only possible to guarantee convergence to the global solution if the breakpoints are chosen in a certain way. The strategies for selecting the breakpoints are described in more detail later on in this section. Additionally, a flowchart illustrating the SGO algorithm is included in Figure 3.

4.1. The preprocessing step. There are many degrees of freedom regarding how to choose the transformations for the nonconvex signomial terms in the MISP problem. For positive terms, the first choice is whether to use the ET or either of the PTs on an individual term, and if either of the PTs is chosen, the transformation power must also be selected. For negative terms, only PTs are applicable, however, also here the transformation power can be selected in infinitely many different ways.

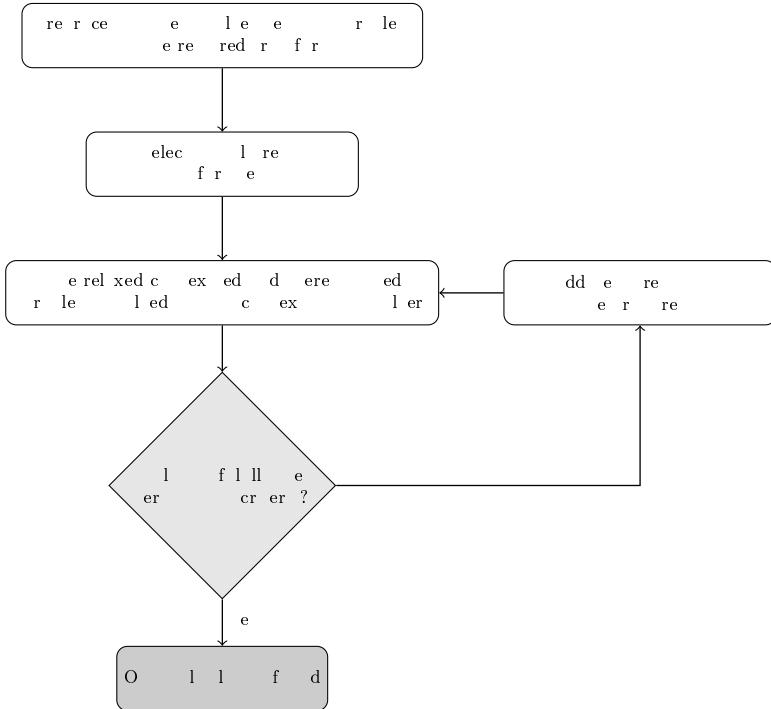


Fig. 3: A flowchart of the SGO algorithm.

Since the goal is to implement the underestimation technique in an automatic manner, a scheme for selecting the transformations in an appropriate manner is needed. In [16], a MILP problem formulation for determining an optimized set of transformations, subject to the values of certain strategy parameters, was presented. It was later extended in [17] to also include the ET for positive signomial terms. Finally, in [20] some additional enhancements were introduced.

By using the MILP method, it is possible to find an optimized set of transformations, subject to certain strategy parameters. For instance, the set containing the smallest number of transformations required to transform the problem or the set of transformations where the fewest original variables in the problems are transformed can be obtained. The latter is important, since it is possible to reuse variables and expressions for the PLFs corresponding to transformations of the same variable, even if the transformations used are not identical, leading to a combinatorially simpler reformulated MISP problem.

4.2. Solving the convex MINLP problem. After the transformations have been determined, the nonconvex signomial constraints are convexified using the single-variable transformations $x_i = T_{ji}(X_{ji})$. Furthermore, the relationships between the inverse transformation and the original variables are expressed using PLFs, for example using the SOS formulation in Section 2.1. After this, the convexified and overestimated MISP problem can, in principle, be solved using any convex MINLP solver.

4.3. Termination criteria. When the solution to the overestimated problem has been found, it is checked whether this solution fulfills the constraints in the original nonconvex problem. Mathematically, this can be stated as an ϵ -criterion, for the $m = 1, \dots, M$ generalized signomial constraints to be satisfied, as

$$\max_m (q_m(\mathbf{x}^*) + \sigma_m(\mathbf{x}^*)) \leq \epsilon_T, \quad (4.1)$$

where \mathbf{x}^* is the optimal value of the current subproblem and $\epsilon_T \geq 0$.

Another ϵ -criterion can additionally be used: If the distance from the solution point \mathbf{x}^* to the nearest breakpoint is less than ϵ_D , no more iterations are performed. This criterion can be specified as

$$\max_i \left(\min_k |x_i^* - \check{x}_{i,k}| \right) \leq \epsilon_D, \quad (4.2)$$

where $\{\check{x}_{i,k}\}_{k=1}^{K_i}$ is the set of breakpoints for the variable x_i and x_i^* is the optimal value for the variable x_i in the current iteration. This criterion is based on the fact that a solution of the transformed problem is exactly equal to the solution given by the original problem at the breakpoints of the PLFs.

4.4. Updating the PLFs. If neither of the termination criteria is met, additional breakpoints must be added to the PLFs. Now there are two things that must be considered, namely which transformation variable approximations to improve and which points to add to the corresponding PLFs? This subject is explained in detail in [31]; here, only a brief summary is given.

The simplest strategy for selecting the variables, is to add breakpoints to the PLFs of all variables transformed. However, for large problems with many different transformations, this may make the transformed problem unnecessary complex. Instead, breakpoints could be added to as few PLFs as possible. For example, it is not necessary to update the PLFs corresponding to transformation variables in nonconvex constraints already fulfilled. A further restriction is to only add breakpoints to the variables in the constraints violated the most in each iteration.

The breakpoints to be added must also be determined. Several strategies exist, for example the solution point in the previous iteration can be added. However, this strategy may, unfortunately, lead to subproblems

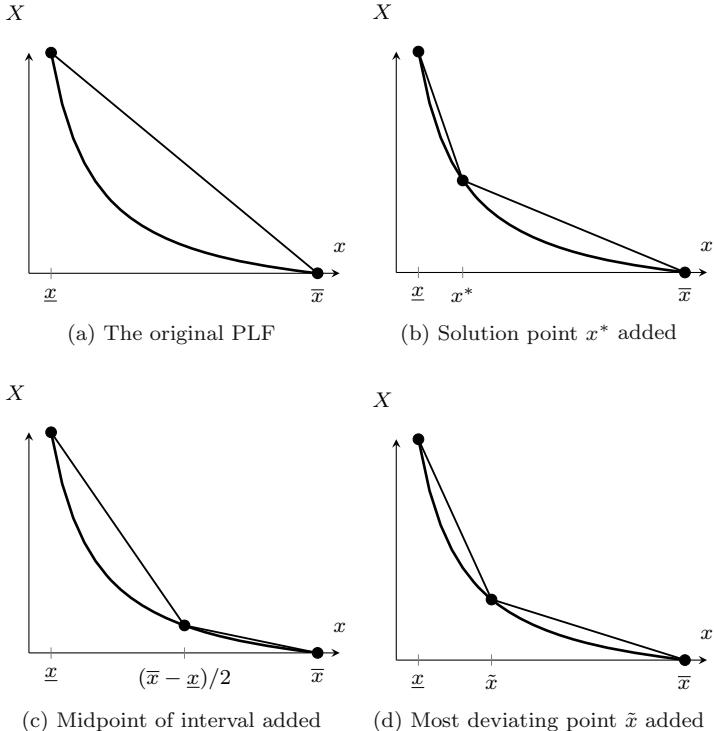


Fig. 4: Illustration of the impact on the PLF approximation when using the different strategies for adding new breakpoints.

whose solution only converges to a local solution of the nonconvex problem. To guarantee to reach the global optimum, other strategies must be used. For example, by adding the midpoint of or the most deviating point in the current interval of breakpoints, to which the previous solution belongs, it has been shown (see [31]) that the global optimal solution will always be found.

The most deviating point for x in the interval $[x, \bar{x}]$ is, according to [15], the point

$$\tilde{x} = \frac{\bar{x} - x}{\ln(\bar{x}/x)} \quad (4.3)$$

for the single-variable ET and

$$\tilde{x} = \left(Q \cdot \frac{\bar{x}^{1/Q} - \underline{x}^{1/Q}}{\bar{x} - \underline{x}} \right)^{\frac{Q}{1-Q}} \quad (4.4)$$

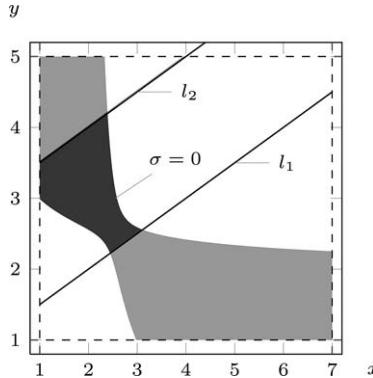


Fig. 5: Shown in the figure are the linear constraints $l_1 : 0.5x_1 - x_2 = 0$ and $l_2 : -0.5x_1 + x_2 = -3$, as well as the signomial constraint $\sigma(x_1, x_2)$. The dark gray region is the integer-relaxed feasible region of the problem.

for any of the single-variable PTs. To exemplify this, an illustration of how the different strategies for selecting the breakpoint improves the PLFs is included in Figure 4.

5. An illustrative example. In this section, the SGO algorithm is applied to the following MISP problem:

$$\begin{aligned}
 & \text{minimize} && -x_1 - 3x_2, \\
 & \text{subject to} && -0.5x_1 + x_2 \leq 3, \\
 & && 0.5x_1 - x_2 \leq 1, \\
 & && \sigma(x_1, x_2) = \sum_{j=1}^8 s_j(x_1, x_2) + 1.5 \leq 0, \\
 & && 1 \leq x_1 \leq 7, \quad 1 \leq x_2 \leq 5, \\
 & && x_1 \in \mathbb{Z}_+, \quad x_2 \in \mathbb{R}_+.
 \end{aligned} \tag{5.1}$$

The signomial terms $s_j(x_1, x_2)$ in the signomial constraint, and the transformations convexifying them, are listed in Table 1. The signomial function $\sigma(x_1, x_2)$ is a modification of the Taylor series expansion of the function $\sin(x_1 + x_2)$ at the point $(x_1, x_2) = (1, 1)$. The feasible region of the original problem is shown in Figure 5.

As can be seen from Table 1, a total number of nine transformations are needed to convexify the nonconvex signomial terms. However, of these transformations, some are identical in different terms, so the number of different transformations are five, three for the variable x_1 and two for x_2 . Furthermore, only two special ordered sets are needed in the PLF

Table 1

The signomial terms in the signomial constraint $\sigma(x_1, x_2)$ in the example in Section 5.

j	$s_j(x_1, x_2)$	$T_j(X_{1,o})$	$T_j(X_{2,o})$	$\hat{s}_j(x_1, x_2, X_{1,o}, X_{2,o})$
1	1.572020	x_1		1.572020
2	-0.435398	x_1^2	$x_1 = X_{1,1}^{0.5}$	-0.435398
3	1.572020	x_2		1.572020
4	-0.832294	$x_1 x_2$	$x_1 = X_{1,1}^{0.5}$	$-0.832294 X_{1,1}^{0.5} X_{2,1}^{0.5}$
5	-0.246575	$x_1^2 x_2$	$x_1 = X_{1,2}^{0.25}$	$-0.246575 X_{1,2}^{0.5} X_{2,1}^{0.5}$
6	-0.435398	x_2^2	$x_2 = X_{2,1}^{0.5}$	-0.435398
7	-0.246575	$x_1 x_2^2$	$x_1 = X_{1,1}^{0.5}$	$-0.246575 X_{1,1}^{0.5} X_{2,2}^{0.5}$
8	0.227324	$x_1^2 x_2^2$	$x_1 = X_{1,3}^{-0.5}$	0.227324

formulation, one for each variable. Thus, the reformulated problem will become

$$\begin{aligned}
 & \text{minimize} && -x_1 - 3x_2, \\
 & \text{subject to} && -0.5x_1 + x_2 \leq 3, \\
 & && 0.5x_1 - x_2 \leq 1, \\
 & && \sum_{j=1}^8 \hat{s}_j(x_1, x_2, \hat{X}_{1,o}, \hat{X}_{2,o}) + 1.5 \leq 0, \\
 & && 1 \leq x_1 \leq 7, \quad 1 \leq x_2 \leq 5, \\
 & && x_1 \in \mathbb{Z}_+, \quad x_2 \in \mathbb{R}_+,
 \end{aligned} \tag{5.2}$$

where the transformation variables $X_{1,o}$ and $X_{2,o}$ have been replaced with the PLFs $\hat{X}_{1,o}$ and $\hat{X}_{2,o}$, which are formulated using the expressions from Section 2.1. The interval endpoints for the variables x_1 and x_2 are used as initial breakpoints for the PLFs. The overestimated feasible region of the problem in the first iteration is illustrated in Figure 7a.

This problem is then solved using a MINLP solver able to solve convex MINLP problems to global optimality. In this case GAMS/ α ECP [32] has been used. The optimal solution in the first iteration is $(x_1, x_2) = (6, 5.00)$, which gives an objective function value of -21.00. The original signomial constraint $\sigma(x_1, x_2)$ has the value 90.48 in this point, so it is not yet optimal since the value is positive. Therefore, more SGO iterations are needed and additional breakpoints must be added in the next iteration. In this example, two different strategies are used: the first one is to add the solution point of the transformed variables, and the second is to add the midpoint of the current interval of breakpoints which the solution belongs

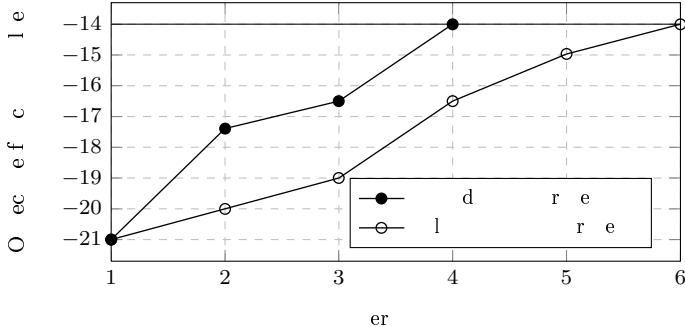


Fig. 6: The objective function value in each SGO iteration.

to. Since x_1 is a discrete variable, the calculated value of the midpoint is rounded upwards if it is noninteger. The solution process and number of SGO iterations required when using the two breakpoint strategies are different as illustrated in Figure 6.

If the midpoint strategy is used, the breakpoint 4 is added to the PLFs $\hat{X}_{1,o}$ and the breakpoint 3 to the PLFs $\hat{X}_{2,o}$ in the second iteration. In total, the solution of this problem required four SGO iterations before the value of the function in the signomial constraint became nonpositive, *i.e.*, the global optimal solution was found. The solution to the overestimated problems in each iteration is listed in Table 2 and the feasible regions are shown in Figure 7.

If instead of adding the midpoint of the interval as a new breakpoint, the solution point itself is added as breakpoint, the solution process will be different. In this case, it will take a total of six iterations to find the global optimal solution. How the feasible region is overestimated in each SGO iteration, is shown in Figure 8.

6. Computational results. A proof-of-concept implementation of the SGO algorithm has been implemented in C# and GAMS [26] and is described in [15] and [19]. Although it is too early to compare it to other available global optimization solvers performance-wise, to give an indication of what sizes of problems the algorithm can solve, it is applied to some trim-loss problems available from the MINLP Library [23]. In this type of problem, all the variables are discrete (binary or integer) and the nonlinearities consists of negative bilinear terms. These problems are somewhat problematic for the solver, since all the variables in the bilinear terms require translations because their lower bounds are zero. Although no additional nonconvex terms appear in the constraints in this type of problems, additional linear terms do. As can be seen from the results in

Table 2

The solution of the convex MINLP problem in each SGO iteration in the example in Section 5 when using the midpoint strategy.

Iter.	Breakpoints		Opt. sol.	x_1	x_2	$\sigma(x_1, x_2)$
	x_1	x_2				
1	{1, 7}	{1, 5}	-21.00	6	5.00	90.48
2	{1, 4, 7}	{1, 3, 5}	-17.39	5	4.13	30.82
3	{1, 4, 6, 7}	{1, 3, 4, 5}	-16.50	3	4.50	5.78
4	{1, 3, 4, 6, 7}	{1, 3, 4, 4.5, 5}	-14.00	2	4.00	-1.72

Table 3

The results of solving some trim-loss problems available in the MINLP Library using an implementation of the SGO algorithm. The number of variables indicated are in the original nonconvex problem. All transformations are power transformations of the type $x_i = X_i^{0.5}$.

Problem	#bin. vars	#int. vars	#transf.	SGO sol.	Glob. sol.
ex1263a	4	20	20	9.3	9.3
ex1264a	4	20	20	8.6	8.6
ex1265a	5	30	30	10.3	10.3
ex1266a	6	42	42	16.3	16.3

Table 3, the globally optimal solution was found in all cases (and all choices of breakpoint and variable selection strategies).

7. Conclusions. In this chapter, the SGO algorithm, a global optimization algorithm for solving nonconvex MISP problems to global optimality as a sequence of convex and overestimated subproblems, was presented. It was shown how the nonconvex problem is reformulated using single-variable transformations applied to the nonconvex signomial terms, after which the relationship describing the inverse transformation between the transformation variables and the original variables are approximated using PLFs. This resulted in a relaxed convex problem, the feasible region of which is an overestimation of that of the original problem. The solution of this transformed problem provides a lower bound to the solution of the original problem, and by iteratively improving the PLFs by adding additional breakpoints, the global optimal solution can be found. It was also illustrated, through an example, how the strategy for selecting the breakpoints impacted the number of iterations required to solve the problem.

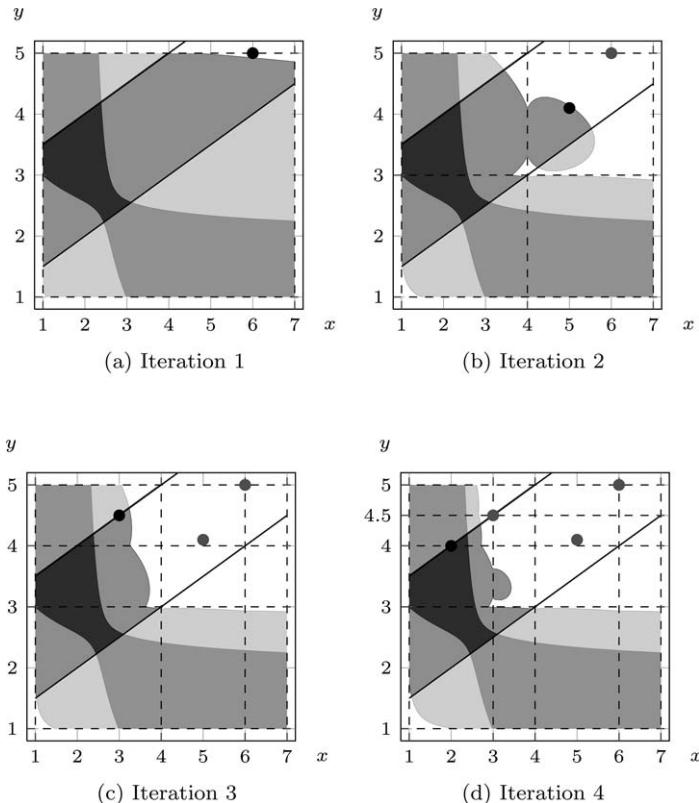


Fig. 7: The feasible region of the convex overestimated problem when using the midpoint strategy. The dark gray region is the integer-relaxed feasible region of the nonconvex problem and the lighter parts correspond to the piecewise convex overestimation of the signomial constraint $\sigma(x_1, x_2)$. The dark points correspond to the optimal solution of the current iteration. The dashed lines indicate the location of the breakpoints in the PLFs.

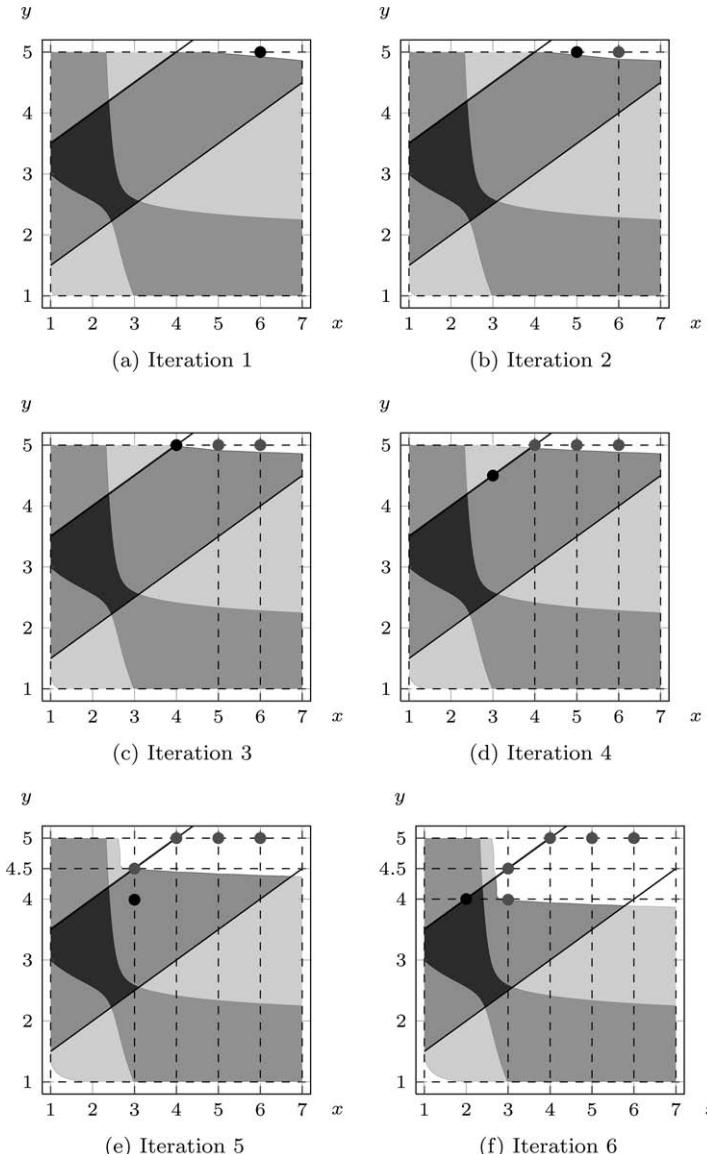


Fig. 8: The feasible region of the convex overestimated problem when using the solution point strategy. The dark gray region is the integer-relaxed feasible region of the nonconvex problem and the lighter parts correspond to the piecewise convex overestimation of the signomial constraint $\sigma(x_1, x_2)$. The dark points correspond to the optimal solution of the current iteration. The dashed lines indicate the location of the breakpoints in the PLFs.

REFERENCES

- [1] C.S. ADJIMAN, S. DALLWIG, C.A. FLOUDAS, AND A. NEUMAIER, *A global optimization method, α BB, for general twice-differentiable constrained NLPs – I. Theoretical advances*, Computers and Chemical Engineering, **22** (1998), pp. 1137–1158.
- [2] M. AVRIEL AND D.J. WILDE, *Optimal condenser design by geometric programming*, Industrial & Engineering Chemistry Process Design and Development, **6** (1967), pp. 256–263.
- [3] E.M.L. BEALE AND J.J.H. FORREST, *Global optimization using special ordered sets*, Mathematical Programming, **10** (1976), pp. 52–69.
- [4] K.-M. BJÖRK, *A Global Optimization Method with Some Heat Exchanger Network Applications*, PhD thesis, bo Akademi University, 2002.
- [5] K.-M. BJÖRK, I. GROSSMANN, AND T. WESTERLUND, *Solving heat exchanger network synthesis problems with non-constant heat capacity flowrates and heat transfer coefficients*, AIDIC Conference Series, **5** (2002), pp. 41–48.
- [6] G.E. BLAU AND D.J. WILDE, *A lagrangian algorithm for equality constrained generalized polynomial optimization*, AIChE Journal, **17** (1971), pp. 235–240.
- [7] R.J. DUFFIN AND E.L. PETERSON, *Duality theory for geometric programming*, SIAM Journal on Applied Mathematics, **14** (1966), pp. 1307–1349.
- [8] C.A. FLOUDAS, *Deterministic Global Optimization. Theory, Methods and Applications*, no. **37** in Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, 1999.
- [9] C.A. FLOUDAS AND P.M. PARDALOS, eds., *Encyclopedia of Optimization*, Kluwer Academic Publishers, 2001.
- [10] I. HARJUNKOSKI, T. WESTERLUND, R. PÖRN, AND H. SKRIFVARS, *Different transformations for solving non-convex trim-loss problems by MINLP*, European Journal of Operational Research, **105** (1998), pp. 594–603.
- [11] Y.H.A. HO, H.-K. KWAN, N. WONG, AND K.-L. HO, *Designing globally optimal delta-sigma modulator topologies via signomial programming*, International Journal of Circuit Theory and Applications, **37** (2009), pp. 453–472.
- [12] R. JABR, *Inductor design using signomial programming*, The International Journal for Computation and Mathematics in Electrical and Electronic Engineering, **26** (2007), pp. 461–475.
- [13] T.R. JEFFERSON AND C.H. SCOTT, *Generalized geometric programming applied to problems of optimal control: I. Theory*, Journal of Optimization Theory and Applications, **26** (1978), pp. 117–129.
- [14] H.-C. LU, H.-L. LI, C.E. GOUNARIS, AND C.A. FLOUDAS, *Convex relaxation for solving posynomial programs*, Journal of Global Optimization, **46f** (2010), pp. 147–154.
- [15] A. LUNDELL, *Transformation Techniques for Signomial Functions in Global Optimization*, PhD thesis, Åbo Akademi University, 2009.
- [16] A. LUNDELL, J. WESTERLUND, AND T. WESTERLUND, *Some transformation techniques with applications in global optimization*, Journal of Global Optimization, **43** (2009), pp. 391–405.
- [17] A. LUNDELL AND T. WESTERLUND, *Exponential and power transformations for convexifying signomial terms in MINLP problems*, in Proceedings of the 27th IASTED International Conference on Modelling, Identification and Control, L. Bruzzone, ed., ACTA Press, 2008, pp. 154–159.
- [18] ———, *Convex underestimation strategies for signomial functions*, Optimization Methods and Software, **24** (2009), pp. 505–522.
- [19] ———, *Implementation of a convexification technique for signomial functions*, in 19th European Symposium on Computer Aided Process Engineering, J. Jezowski and J. Thullie, eds., Vol. **26** of Computer Aided Chemical Engineering, Elsevier, 2009, pp. 579–583.

- [20] ———, *Optimization of transformations for convex relaxations of MINLP problems containing signomial functions*, in Proceedings 10th International Symposium on Process Systems Engineering (PSE2009), 2009.
- [21] C.D. MARANAS AND C.A. FLOUDAS, *Finding all solutions of nonlinearly constrained systems of equations*, Journal of Global Optimization, **7** (1995), pp. 143–182.
- [22] G.P. MCCORMICK, *Mathematical programming computability of global solutions to factorable nonconvex programs: Part I – convex underestimating problems*, Mathematical Programming, **10** (1976), pp. 147–175.
- [23] MINLPWORLD, *The MINLP Library*, <http://www.gamsworld.org/minlp/>.
- [24] U. PASSY AND D.J. WILDE, *A geometric programming algorithm for solving chemical equilibrium problems*, SIAM Journal on Applied Mathematics, **16** (1968), pp. 363–373.
- [25] R. PÖRN, *Mixed Integer Non-Linear Programming: Convexification Techniques and Algorithm Development*, PhD thesis, bo Akademi University, 2000.
- [26] R. E. ROSENTHAL, *GAMS – A user’s guide*, GAMS Development Corporation, Washington, DC, USA, 2008.
- [27] N. V. SAHINIDIS AND M. TAWARMALANI, *BARON 7.2.5: Global optimization of mixed-integer nonlinear programs, user’s manual*, 2005.
- [28] Y. SHEN, E. Y. LAM, AND N. WONG, *Binary image restoration by signomial programming*, in OSA Topical Meeting in Signal Recovery and Synthesis, Optical Society of America, 2007.
- [29] M. TAWARMALANI AND N.V. SAHINIDIS, *Global optimization of mixed-integer nonlinear programs: A theoretical and computational study*, Mathematical Programming, **99** (2004), pp. 563–591.
- [30] J.F. TSAI AND M.-H. LIN, *Global optimization of signomial mixed-integer nonlinear programming problems with free variables*, Journal of Global Optimization, **42** (2008), pp. 39–49.
- [31] T. WESTERLUND, *Some transformation techniques in global optimization*, in Global Optimization: From Theory to Implementation, L. Liberti and N. Maculan, eds., Vol. **84** of Nonconvex Optimization and its Applications, Springer, 2005, pp. 47–74.
- [32] T. WESTERLUND AND T. LASTUSILTA, *AlphaECP GAMS user’s manual*, Åbo Akademi University, 2008.
- [33] T. WESTERLUND AND J. WESTERLUND, *GGPECP – An algorithm for solving non-convex MINLP problems by cutting plane and transformation techniques*, Chemical Engineering Transactions, **3** (2003), pp. 1045–1050.

PART VI:

MIXED-INTEGER QUADRATICALLY
CONSTRAINED OPTIMIZATION

THE MILP ROAD TO MIQCP

SAMUEL BURER* AND ANUREET SAXENA†

Abstract. This paper surveys results on the NP-hard mixed-integer quadratically constrained programming problem. The focus is strong convex relaxations and valid inequalities, which can become the basis of efficient global techniques. In particular, we discuss relaxations and inequalities arising from the algebraic description of the problem as well as from dynamic procedures based on disjunctive programming. These methods can be viewed as generalizations of techniques for mixed-integer linear programming. We also present brief computational results to indicate the strength and computational requirements of these methods.

1. Introduction. More than fifty years have passed since Dantzig et al. [25] solved the 50-city travelling salesman problem. An achievement in itself at the time, their seminal paper gave birth to one of the most successful disciplines in computational optimization, Mixed Integer Linear Programming (MILP). Five decades of wonderful research, both theoretical and computational, have brought mixed integer programming to a stage where it can solve many if not all MILPs arising in practice (see [43]). The ideas discovered during the course of this development have naturally influenced other disciplines. Constraint programming, for instance, has adopted and refined many of the ideas from MILP to solve more general classes of problems [2].

Our focus in this paper is to track the influence of MILP in solving mixed integer quadratically constrained problems (MIQCP). In particular, we survey some of the recent research on MIQCP and establish their connections to well known ideas in MILP. The purpose of this is two-fold. First, it helps to catalog some of the recent results in a form that is accessible to a researcher with reasonable background in MILP. Second, it defines a roadmap for further research in MIQCP; although significant progress has been made in the field of MIQCP, the “breakthrough” results are yet to come and we believe that the past of MILP holds the clues to the future of MIQCP.

Specifically, we focus on the following mixed integer quadratically constrained problem

$$\begin{aligned} \min \quad & x^T Cx + c^T x \\ \text{s.t.} \quad & x \in \mathcal{F} \end{aligned} \tag{MIQCP}$$

*Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1994, USA (samuel-burer@uiowa.edu). Author supported in part by NSF Grant CCF-0545514.

† Axioma Inc., 8800 Roswell Road, Building B. Suite 295, Atlanta GA, 30350, USA (asaxena@axiomainc.com).

where

$$\mathcal{F} := \left\{ x \in \mathbb{R}^n : \begin{array}{l} x^T A_k x + a_k^T x \leq b_k \quad \forall k = 1, \dots, m \\ l \leq x \leq u \\ x_i \in \mathbb{Z} \quad \forall i \in I \end{array} \right\}.$$

The data of **(MIQCP)** is

- $(C, c) \in \mathcal{S}^n \times \mathbb{R}^n$
- $(A_k, a_k, b_k) \in \mathcal{S}^n \times \mathbb{R}^n \times \mathbb{R}$ for all $k = 1, \dots, m$
- $(l, u) \in (\mathbb{R} \cup \{-\infty\})^n \times (\mathbb{R} \cup \{+\infty\})^n$
- $I \subseteq [n]$

where, in particular, \mathcal{S}^n is the set of all $n \times n$ symmetric matrices and $[n] := \{1, \dots, n\}$. Without loss of generality, we assume $l < u$, and for all $i \in I$, $(l_i, u_i) \in (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{+\infty\})$. Note that any specific lower or upper bound may be infinite.

If all $A_k = 0$, then **(MIQCP)** reduces to MILP. So **(MIQCP)** is NP-hard. In fact, the continuous variant of MIQCP, namely a non-convex QCP, is already NP-hard and a well-known problem in global optimization [45, 46]. The computational intractability of MIQCP is quite notorious and can be traced to the result of Jeroslow [30] from the seventies that shows that the variant of MIQCP without explicit non-infinite lower/upper bounds on some of the variables is undecidable. **(MIQCP)** is itself a special case of mixed integer nonlinear programming (MINLP); we refer the reader to the website *MINLP World* [41] for surveys, software, and test instances for MINLP. We also note that any polynomial optimization problem may be reduced to **(MIQCP)** by the introduction of auxiliary variables and constraints to reduce all polynomial degrees to 2, e.g., a cubic term $x_1 x_2 x_3$ could be modeled as $x_1 X_{23}$ with $X_{23} = x_2 x_3$.

Note that if the objective function and constraints in MIQCP are convex, then the resulting optimization problem can be solved using standard techniques for solving convex MINLP (see [16] for more details). Most of the ideas and methods discussed in this paper specifically exploit the non-convex quadratic nature of the objective and constraints of **(MIQCP)**. In fact, our viewpoint is that many ideas from the solution of MILPs can be adapted in interesting ways for the study of **(MIQCP)**. In this sense, we view **(MIQCP)** as a natural progression from MILP rather than, say, a special case of MINLP.

We are also not specifically concerned with the global optimization of **(MIQCP)**. Rather, we focus on generating strong convex relaxations and valid inequalities, which could become the basis of efficient global techniques.

In Section 2, we review the idea of *lifting*, which is commonly used to convexify **(MIQCP)** and specifically the feasible set \mathcal{F} . We then discuss the generation of various types of linear, second-order-cone, and semidefinite valid inequalities which strengthen the convexification. These inequalities have the property that they arise directly from the algebraic form of \mathcal{F} .

In this sense, they generalize the basic LP relaxation often used in MILP. We also catalog several known and new results establishing the strength of these inequalities for certain specifications of \mathcal{F} . Then, in Section 3, we describe several related approaches that shed further light on convex relaxations of **(MIQCP)**.

In Section 4, we discuss methods for dynamically generating valid inequalities, which can further improve the relaxations. One of the fundamental tools is that of disjunctive programming, which has been used in the MILP community for five decades. However, the disjunctions employed herein are new in the sense that they truly exploit the quadratic form of **(MIQCP)**. Recently, Belotti [11] studies disjunctive cuts for general MINLP.

Finally, in Section 5, we consider a short computational study to give some sense of the computational effort and effect of the methods surveyed in this paper.

1.1. Notation and terminology. Most of the notation used in this paper is standard. We define here just a few perhaps atypical notations. For symmetric matrices A and B of conformable dimensions, we define $\langle A, B \rangle = \text{tr}(AB)$; a standard fact is that the quadratic form $x^T Ax$ can be represented as $\langle A, xx^T \rangle$. For a set P in the space of variables (x, y) , $\text{proj}_x(P)$ denotes the coordinate projection of P onto the space x . $\text{clconv } P$ is the closed convex hull of P . For a square matrix A , $\text{diag}(A)$ denotes the vector of diagonal entries of A . The vector e is the all-ones vector, and e_i is the vector having all zeros except a 1 in position i . The notation $X \succeq 0$ means that X is symmetric positive semidefinite; $X \preceq 0$ means symmetric negative semidefinite.

2. Convex relaxations and valid inequalities. In this section, we describe strong convex relaxations of **(MIQCP)** and \mathcal{F} , which arise from the algebraic description of \mathcal{F} . For the purposes of presentation, we partition the indices $[m]$ of the quadratic constraints into three groups:

$$\begin{aligned} \text{"linear"} \quad LQ &:= \{k : A_k = 0\} \\ \text{"convex quadratic"} \quad CQ &:= \{k : A_k \neq 0, A_k \succeq 0\} \\ \text{"nonconvex quadratic"} \quad NQ &:= \{k : A_k \neq 0, A_k \not\succeq 0\}. \end{aligned}$$

For those $k \in CQ$, there exists a rectangular matrix B_k (not necessarily unique) such that $A_k = B_k^T B_k$. Using the B_k 's, it is well known that each convex quadratic constraint can be represented as a second-order-cone constraint.

PROPOSITION 2.1 (Alizadeh and Goldfarb [5]). *Let $k \in CQ$ with $A_k = B_k^T B_k$. A point $x \in \mathbb{R}^n$ satisfies $x^T A_k x + a_k^T x \leq b_k$ if and only if*

$$\left\| \begin{pmatrix} B_k x \\ \frac{1}{2}(1 + a_k^T x - b_k) \end{pmatrix} \right\| \leq \frac{1}{2}(1 - a_k^T x + b_k).$$

So $\mathcal{F} \subseteq \mathbb{R}^n$ may be rewritten as

$$\mathcal{F} = \left\{ x : \begin{array}{l} a_k^T x \leq b_k \quad \forall k \in LQ \\ \left\| \begin{pmatrix} B_k^T x \\ \frac{1}{2}(a_k^T x - b_k + 1) \end{pmatrix} \right\| \leq \frac{1}{2}(1 - a_k^T x + b_k) \quad \forall k \in CQ \\ x^T A_k x + a_k^T x \leq b_k \quad \forall k \in NQ \\ l \leq x \leq u \\ x_i \in \mathbb{Z} \quad \forall i \in I \end{array} \right\}.$$

If so desired, we can model the bounds $l \leq x \leq u$ within the linear constraints. However, since bounds often play a special role in approaches for **(MIQCP)**, we leave them separate.

2.1. Lifting, convexification, and relaxation. An idea fundamental to many methods for **(MIQCP)** is *lifting* to a higher dimensional space. The simplest lifting idea is to introduce auxiliary variables X_{ij} , which model the quadratic terms $x_i x_j$ via equations $X_{ij} = x_i x_j$ for all $1 \leq i, j \leq n$. The single symmetric matrix equation $X = xx^T$ also captures this lifting.

As an immediate consequence of lifting, the quadratic objective and constraints may be expressed linearly in (x, X) , e.g.,

$$x^T C x + c^T x \stackrel{X=xx^T}{=} \langle C, X \rangle + c^T x.$$

So **(MIQCP)** becomes

$$\begin{aligned} \min \quad & \langle C, X \rangle + c^T x \\ \text{s.t.} \quad & (x, X) \in \widehat{\mathcal{F}} \end{aligned} \tag{2.1}$$

where

$$\widehat{\mathcal{F}} := \left\{ (x, X) \in \mathbb{R}^n \times \mathcal{S}^n : \begin{array}{l} \langle A_k, X \rangle + a_k^T x \leq b_k \quad \forall k = 1, \dots, m \\ l \leq x \leq u \\ x_i \in \mathbb{Z} \quad \forall i \in I \\ X = xx^T \end{array} \right\}.$$

This provides an interesting perspective: the “hard” quadratic objective and constraints of **(MIQCP)** are represented as “easy” linear ones in the space (x, X) . The trade-off is the nonconvex equation $X = xx^T$, and of course the non-convex integrality conditions remain.

The linear objective in (2.1) allows convexification of the feasible region without change to the optimal value. From standard convex optimization:

PROPOSITION 2.2. *The problem (2.1), and hence also **(MIQCP)**, is equivalent to*

$$\min \left\{ \langle C, X \rangle + c^T x : (x, X) \in \text{clconv } \widehat{\mathcal{F}} \right\}.$$

Thus, many lifting approaches may be interpreted as attempting to better understand $\text{clconv} \widehat{\mathcal{F}}$. We adopt this point of view.

A straightforward linear relaxation of $\text{clconv} \widehat{\mathcal{F}}$, which is analogous to the basic linear relaxation of a MILP, is gotten by simply dropping $X = xx^T$ and $x_i \in \mathbb{Z}$:

$$\widehat{\mathcal{L}} := \left\{ (x, X) \in \mathbb{R}^n \times \mathcal{S}^n : \begin{array}{l} \langle A_k, X \rangle + a_k^T x \leq b_k \quad \forall k = 1, \dots, m \\ l \leq x \leq u \end{array} \right\}.$$

There are many ways to strengthen $\widehat{\mathcal{L}}$ as discussed in the following three subsections.

2.2. Valid linear inequalities. The most common idea for constructing valid linear inequalities for $\text{clconv} \widehat{\mathcal{F}}$ is the following. Let $\alpha^T x \leq \alpha_0$ and $\beta^T x \leq \beta_0$ be any two valid linear inequalities for \mathcal{F} (possibly the same). Then the quadratic inequality

$$0 \leq (\alpha_0 - \alpha^T x)(\beta_0 - \beta^T x) = \alpha_0 \beta_0 - \beta_0 \alpha^T x - \alpha_0 \beta^T x + x^T \alpha \beta^T x$$

is also valid for \mathcal{F} , and so the linear inequality

$$\alpha_0 \beta_0 - \beta_0 \alpha^T x - \alpha_0 \beta^T x + \langle \beta \alpha^T, X \rangle \geq 0$$

is valid for $\text{clconv} \widehat{\mathcal{F}}$.

The above idea works with *any* pair of valid inequalities, e.g., ones given explicitly in the description of \mathcal{F} or derived ones. For those explicitly given (the bounds $l \leq x \leq u$ and the constraints corresponding to $k \in LQ$), the complete list of derived valid quadratic inequalities is

$$\left. \begin{array}{l} (x_i - l_i)(x_j - l_j) \geq 0 \\ (x_i - l_i)(u_j - x_j) \geq 0 \\ (u_i - x_i)(x_j - l_j) \geq 0 \\ (u_i - x_i)(u_j - x_j) \geq 0 \end{array} \right\} \quad \forall (i, j) \in [n] \times [n], i \leq j \quad (2.2a)$$

$$\left. \begin{array}{l} (x_i - l_i)(b_k - a_k^T x) \geq 0 \\ (u_i - x_i)(b_k - a_k^T x) \geq 0 \end{array} \right\} \quad \forall (i, k) \in [n] \times LQ \quad (2.2b)$$

$$(b_\ell - a_\ell^T x)(b_k - a_k^T x) \geq 0 \quad \forall (\ell, k) \in LQ \times LQ, \ell \leq k. \quad (2.2c)$$

The linearizations of (2.2) were considered in [37] and are sometimes referred to as *rank-2 linear inequalities* [33]. So we denote the collection of all (x, X) , which satisfy these linearizations, as R_2 , i.e.,

$$R_2 := \{ (x, X) : \text{linearized versions of (2.2) hold} \}.$$

In particular, the linearized versions of (2.2a) are called the RLT inequalities after the “reformulation-linearization technique” of [55], though they first appeared in [3, 4, 40]. These inequalities have been studied extensively because of their wide applicability and simple structure. Specifically,

the RLT constraints provide simple bounds on the entries of X , which otherwise may be weakly constrained in $\widehat{\mathcal{L}}$, especially when m is small. After linearization via $X_{ij} = x_i x_j$, the four inequalities (2.2a) for a specific (i, j) are

$$\left\{ \begin{array}{l} l_i x_j + x_i l_j - l_i l_j \\ u_i x_j + x_i u_j - u_i u_j \end{array} \right\} \leq X_{ij} \leq \left\{ \begin{array}{l} x_i u_j + l_i x_j - l_i u_j \\ x_i l_j + u_i x_j - u_i l_j \end{array} \right. \quad (2.3)$$

Using the symmetry of X , the entire collection of RLT inequalities in matrix form is

$$\left\{ \begin{array}{l} l x^T + x l^T - l l^T \\ u x^T + x u^T - u u^T \end{array} \right\} \leq X \leq x u^T + l x^T - l u^T. \quad (2.4)$$

It can be shown easily that the original inequalities $l \leq x \leq u$ are implied by (2.4) if both l and u are finite in all components.

Since the RLT inequalities are just a portion of the inequalities defining R_2 , it might be reasonable to consider R_2 generally and not the RLT constraints specifically. However, it is sometimes convenient to study the RLT constraints on their own. So we write $(x, X) \in \text{RLT}$ when (x, X) satisfies (2.4) but not necessarily all rank-2 linear inequalities, i.e.,

$$\text{RLT} := \{ (x, X) : (2.4) \text{ holds} \}.$$

2.3. Valid second-order-cone inequalities. Similar to the derivation of the inequalities defining R_2 , the linearizations of the following quadratic inequalities are valid for $\text{clconv } \widehat{\mathcal{F}}$: for all $(k, \ell) \in LQ \times CQ$,

$$(b_k - a_k^T x) \left(\frac{1}{2}(1 - a_\ell^T x + b_\ell) - \left\| \begin{pmatrix} B_\ell^T x \\ \frac{1}{2}(a_\ell^T x - b_\ell + 1) \end{pmatrix} \right\| \right) \geq 0. \quad (2.5)$$

We call the linearizations *rank-2 second-order inequalities* and denote by S_2 the set of all satisfying (x, X) , i.e.,

$$S_2 := \{ (x, X) : \text{linearized versions of (2.5) hold} \}.$$

As an example, suppose we have the two valid inequalities $x_1 + x_2 \leq 1$ and $x_1^2 + x_2^2 \leq 2/3$, the second of which, via Proposition 2.1, is equivalent to the second-order cone constraint $\|x\| \leq \sqrt{2/3}$. Then we multiply $1 - x_1 - x_2 \geq 0$ with $\sqrt{2/3} - \|x\| \geq 0$ to obtain the valid inequality

$$\begin{aligned} 0 &\leq (1 - x_1 - x_2)(\sqrt{2/3} - \|x\|) \\ &= \sqrt{2/3}(1 - x_1 - x_2) - \|(1 - x_1 - x_2)x\|, \end{aligned}$$

which is linearized as

$$\left\| \begin{pmatrix} x_1 - X_{11} - X_{12} \\ x_2 - X_{21} - X_{22} \end{pmatrix} \right\| \leq \sqrt{2/3}(1 - x_1 - x_2).$$

2.4. Valid semidefinite inequalities. The application of SDP to **(MIQCP)** arises from the following fundamental observation:

LEMMA 2.1 (Shor [56]). *Given $x \in \mathbb{R}^n$, it holds that*

$$\begin{pmatrix} 1 & x^T \\ x & xx^T \end{pmatrix} = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^T \succeq 0.$$

Thus, the linearized semidefinite inequality

$$Y := \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 \quad (2.6)$$

is valid for $\text{clconv } \widehat{\mathcal{F}}$. We define

$$\text{PSD} := \{ (x, X) : (2.6) \text{ holds} \}.$$

Instead of enforcing $(x, X) \in \text{PSD}$, i.e., the full PSD condition (2.6), one can enforce relaxations of it. For example, since all principal submatrices of $Y \succeq 0$ are semidefinite, one could enforce just that all or some of the 2×2 principal submatrices of Y are positive semidefinite. This has been done in [32], for example.

2.5. The strength of valid inequalities. From the preceding subsections, we have the following result by construction:

PROPOSITION 2.3. $\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{L}} \cap \text{RLT} \cap R_2 \cap S_2 \cap \text{PSD}$.

Even though $R_2 \subseteq \text{RLT}$, we retain RLT in the above expression for emphasis. We next catalog and discuss various special cases in which equality is known to hold in Proposition 2.3.

2.5.1. Simple bounds. We first consider the case when \mathcal{F} is defined by simple, finite bounds, i.e., $\mathcal{F} = \{x \in \mathbb{R}^n : l \leq x \leq u\}$ with (l, u) finite in all components. In this case, $R_2 = \text{RLT} \subseteq \widehat{\mathcal{L}}$ and S_2 is vacuous. So Proposition 2.3 can be stated more simply as $\text{clconv } \widehat{\mathcal{F}} \subseteq \text{RLT} \cap \text{PSD}$. Equality holds if and only if $n \leq 2$:

THEOREM 2.1 (Anstreicher and Burer [6]). *Let $\mathcal{F} = \{x \in \mathbb{R}^n : l \leq x \leq u\}$ with (l, u) finite in all components. Then $\text{clconv } \widehat{\mathcal{F}} \subseteq \text{RLT} \cap \text{PSD}$ with equality if and only if $n \leq 2$.*

For $n > 2$, [6] and [19] derive additional valid inequalities but are still unable to determine an exact representation by valid inequalities even for $n = 3$. ([6] does give an exact *disjunctive* representation for $n = 3$.)

We also mention a classical result, which is in some sense subsumed by Theorem 2.1. Even still, this result indicates the strength of the RLT inequalities and can be useful when one-variable quadratics $X_{ii} = x_i^2$ are not of interest. The result does not fully classify $\text{clconv } \widehat{\mathcal{F}}$ but rather coordinate projections of it.

THEOREM 2.2 (Al-Khayyal and Falk [4]). *Let $\mathcal{F} = \{x \in \mathbb{R}^n : l \leq x \leq u\}$ with (l, u) finite in all components. Then, for all $1 \leq i < j \leq n$,*

$\text{proj}_{(x_i, x_j, X_{ij})}(\text{clconv } \widehat{\mathcal{F}}) = \text{RLT}_{ij}$, where $\text{RLT}_{ij} := \{(x_i, x_j, X_{ij}) \in \mathbb{R}^3 : (2.3) \text{ holds}\}$.

2.5.2. Binary integer grid. We next consider the case when \mathcal{F} is a binary integer grid: that is, $\mathcal{F} = \{x \in \mathbb{Z}^n : l \leq x \leq u\}$ with $u = l + e$ and l finite in all components. Note that this is simply a shift of the standard 0-1 binary grid and that $\text{clconv } \widehat{\mathcal{F}}$ is a polytope. In this case, $R_2 = \text{RLT} \subseteq \widehat{\mathcal{L}}$ and S_2 is vacuous. So Proposition 2.3 states that $\text{clconv } \widehat{\mathcal{F}} \subseteq \text{RLT} \cap \text{PSD}$. Also, some additional, simple linear equations are valid for $\text{clconv } \widehat{\mathcal{F}}$.

PROPOSITION 2.4. *Suppose that $i \in I$ has $u_i = l_i + 1$ with l_i finite. Then the equation $X_{ii} = (1 + 2l_i)x_i - l_i - l_i^2$ is valid for $\text{clconv } \widehat{\mathcal{F}}$.*

Proof. The shift $x_i - l_i$ is either 0 or 1. Hence, $(x_i - l_i)^2 = x_i - l_i$. After linearization with $X_{ii} = x_i^2$, this quadratic equation becomes the claimed linear one. \square

When $I = [n]$, the individual equations $X_{ii} = (1 + 2l_i)x_i - l_i - l_i^2$ can be collected as $\text{diag}(X) = (e + 2l) \circ x - l - l^2$. We remark that the next result does not make use of PSD.

THEOREM 2.3 (Padberg [44]). *Let $\mathcal{F} = \{x \in \mathbb{Z}^n : l \leq x \leq u\}$ with $u = l + e$ and l finite in all components. Then*

$$\text{clconv } \widehat{\mathcal{F}} \subseteq \text{RLT} \cap \{(x, X) : \text{diag}(X) = (e - 2l) \circ x - l - l^2\}$$

with equality if and only if $n \leq 2$.

In this case, $\text{clconv } \widehat{\mathcal{F}}$ is closely related to the *boolean quadric polytope* [44]. For $n > 2$, additional valid inequalities, such as the triangle inequalities described by Padberg [44], provide an even better approximation of $\text{clconv } \widehat{\mathcal{F}}$. For general n , a full description should not be easily available unless $P = NP$.

2.5.3. The nonnegative orthant and standard simplex. We now consider a case arising in the study of completely positive matrices in optimization and linear algebra [13, 23, 24]. A matrix Y is *completely positive* if $Y = NN^T$ for some nonnegative, rectangular matrix N , and the set of all completely positive matrices is a closed, convex cone. Although this cone is apparently intractable [42], it can be approximated from the outside to any precision using a sequence of polyhedral-semidefinite relaxations [26, 47]. The simplest approximation is by the so-called *doubly nonnegative matrices*, which are matrices Y satisfying $Y \succeq 0$ and $Y \geq 0$. Clearly, every completely positive matrix Y is doubly nonnegative. The converse holds if and only if $n \leq 4$ [39].

Let $\mathcal{F} = \{x \in \mathbb{R}^n : x \geq 0\}$. Then, since $\text{RLT} = R_2$ and S_2 is vacuous, Proposition 2.3 states $\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$. Note that, because u is infinite in this case, $(x, X) \in \text{RLT}$ does not imply $x \geq 0$, and so we must include $\widehat{\mathcal{L}}$ explicitly to enforce $x \geq 0$. It is easy to see that

$$\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD} = \left\{ (x, X) \geq 0 : \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 \right\} \quad (2.7)$$

which is the set of all doubly nonnegative matrices of size $(n+1) \times (n+1)$ having a 1 in the top-left corner.

For general n , it appears that equality in Proposition 2.3 does not hold in this case. However, it does hold for $n \leq 3$, which we prove next. As far as we are aware, this result has not appeared in the literature.

We first characterize the difference between $\text{conv } \widehat{\mathcal{F}}$ and $\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$ for $n \leq 3$. The following lemma shows that this difference is precisely the recession cone of $\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$, which equals

$$\text{rcone}(\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}) = \{(0, D) \geq 0 : D \succeq 0\}. \quad (2.8)$$

LEMMA 2.2. *Let $\mathcal{F} = \{x \in \mathbb{R}^n : x \geq 0\}$ with $n \leq 3$. Then*

$$\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD} = \text{conv } \widehat{\mathcal{F}} + \text{rcone}(\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}).$$

Proof. To prove the statement, we show containment in both directions; the containment \supseteq is easy. To show \subseteq , let $(x, X) \in \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$ be arbitrary. By (2.7),

$$Y := \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix}$$

is doubly nonnegative of size $(n+1) \times (n+1)$. Since $n \leq 3$, the “ $n \leq 4$ ” result of [39] implies that Y is completely positive. Hence, there exists a rectangular $N \geq 0$ such that $Y = NN^T$. By decomposing each column $N_{\cdot j}$ of N as

$$N_{\cdot j} = \begin{pmatrix} \zeta_j \\ z_j \end{pmatrix}, \quad (\zeta_j, z_j) \in \mathbb{R}_+ \times \mathbb{R}_+^n$$

we can write

$$\begin{aligned} Y &= \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} = \sum_j \begin{pmatrix} \zeta_j \\ z_j \end{pmatrix} \begin{pmatrix} \zeta_j \\ z_j \end{pmatrix}^T \\ &= \sum_{j: \zeta_j > 0} \zeta_j^2 \begin{pmatrix} 1 \\ \zeta_j^{-1} z_j \end{pmatrix} \begin{pmatrix} 1 \\ \zeta_j^{-1} z_j \end{pmatrix}^T + \sum_{j: \zeta_j = 0} \begin{pmatrix} 0 \\ z_j \end{pmatrix} \begin{pmatrix} 0 \\ z_j \end{pmatrix}^T \\ &= \sum_{j: \zeta_j > 0} \zeta_j^2 \begin{pmatrix} 1 \\ \zeta_j^{-1} z_j & \zeta_j^{-1} z_j^T \\ \zeta_j^{-2} z_j z_j^T & z_j z_j^T \end{pmatrix} + \sum_{j: \zeta_j = 0} \begin{pmatrix} 0 & 0^T \\ 0 & z_j z_j^T \end{pmatrix}. \end{aligned}$$

This shows that $\sum_{j: \zeta_j > 0} \zeta_j^2 = 1$ and, from (2.8), that (x, X) is expressed as the convex combination of points in $\widehat{\mathcal{F}}$ plus the sum of points in $\text{rcone}(\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD})$, as desired. \square

Using the lemma, we can prove equality in Proposition 2.3 for $n \leq 3$.

THEOREM 2.4. *Let $\mathcal{F} = \{x \in \mathbb{R}^n : x \geq 0\}$. Then $\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$ with equality if $n \leq 3$.*

Proof. The first statement of the theorem is just Proposition 2.3. Next, for contradiction, suppose there exists $(\bar{x}, \bar{X}) \in \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD} \setminus \text{clconv } \widehat{\mathcal{F}}$. By the separating hyperplane theorem, there exists (c, C) such that

$$\min\{\langle C, X \rangle + c^T x : x \in \text{clconv } \widehat{\mathcal{F}}\} \geq 0 > \langle C, \bar{X} \rangle + c^T \bar{x}.$$

Since $(\bar{x}, \bar{X}) \in \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$, by the lemma there exists $(z, Z) \in \text{conv } \widehat{\mathcal{F}}$ and $(0, D) \in \text{rcone}(\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD})$ such that $(\bar{x}, \bar{X}) = (z, Z + D)$. Thus, $\langle C, D \rangle < 0$.

Since $D \geq 0$, $D \succeq 0$, and $n \leq 3$, D is completely positive, i.e., there exists rectangular $N \geq 0$ such that $D = NN^T$. We have $\langle C, NN^T \rangle < 0$, which implies $d^T Cd < 0$ for some nonzero column $d \geq 0$ of N . It follows that d is a negative direction of recession for the function $x^T Cx + c^T x$. In other words,

$$\min\{\langle C, X \rangle + c^T x : x \in \text{clconv } \widehat{\mathcal{F}}\} = -\infty,$$

a contradiction. \square

A related result occurs for a bounded slice of the nonnegative orthant, e.g., the standard simplex $\{x \geq 0 : e^T x = 1\}$. In this case, however, the boundedness, the linear constraint, and R_2 ensure that equality holds in Proposition 2.3 for $n \leq 4$.

THEOREM 2.5 (Anstreicher and Burer [6]). *Let $\mathcal{F} := \{x \geq 0 : e^T x = 1\}$. Then $\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{L}} \cap \text{RLT} \cap R_2 \cap \text{PSD}$ with equality if and only if $n \leq 4$.*

[36] and [6] also give related results where \mathcal{F} is an affine transformation of the standard simplex.

2.5.4. Half ellipsoid. Let \mathcal{F} be a half ellipsoid, that is, the intersection of a linear half-space and a possibly degenerate ellipsoid. In contrast to the previous cases considered, [57] proved that this case achieves equality in Proposition 2.3 regardless of the dimension n . On the other hand, the number of constraints is fixed. In particular, all simple bounds are infinite, $|LQ| = 1$, $|CQ| = 1$, and $NQ = \emptyset$ in which case Proposition 2.3 states simply $\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{L}} \cap S_2 \cap \text{PSD}$.

THEOREM 2.6 (Sturm and Zhang [57]). *Suppose*

$$\mathcal{F} = \left\{ x \in \mathbb{R}^n : \begin{array}{l} a_1^T x \leq b_1 \\ x^T A_2 x + a_2^T x \leq b_2 \end{array} \right\}$$

with $A_2 \succeq 0$ is nonempty. Then $\text{clconv } \widehat{\mathcal{F}} = \widehat{\mathcal{L}} \cap S_2 \cap \text{PSD}$.

As far as we are aware, this is the only case where Proposition 2.3 is provably strengthened via use of the rank-2 second-order inequalities enforced by S_2 .

2.5.5. Bounded quadratic form. The final case we consider is that of a bounded quadratic form. Specifically, for a given quadratic form $x^T Ax + a^T x$ and bounds $-\infty \leq b_l \leq b_u \leq +\infty$, let \mathcal{F} be the set of points such that the form falls within the bounds, i.e., $\mathcal{F} = \{x : b_l \leq x^T Ax + a^T x \leq b_u\}$. No assumptions are made on A , e.g., we do not assume that A is positive semidefinite. As far as we are aware, the result proved below is new, but closely related results can be found in [57, 61].

Since there are no explicit bounds and no linear or convex quadratic inequalities, Proposition 2.3 states simply that $\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{L}} \cap \text{PSD}$, where

$$\widehat{\mathcal{L}} \cap \text{PSD} = \left\{ (x, X) : \begin{array}{l} b_l \leq \langle A, X \rangle + a^T x \leq b_u \\ \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 \end{array} \right\}. \quad (2.9)$$

In general, it appears that equality in Proposition 2.3 does not hold in this case. However, we can still characterize the difference between $\text{clconv } \widehat{\mathcal{F}}$ and $\widehat{\mathcal{L}} \cap \text{PSD}$. As it turns out in the theorem below, this difference is precisely the recession cone of $\widehat{\mathcal{L}} \cap \text{PSD}$, which equals

$$\text{rcone}(\widehat{\mathcal{L}} \cap \text{PSD}) = \left\{ (0, D) : \begin{array}{ll} 0 \leq \langle A, D \rangle & \text{if } -\infty < b_l \\ \langle A, D \rangle \leq 0 & \text{if } b_u < +\infty \\ D \succeq 0. & \end{array} \right\}.$$

In particular, if $\text{rcone}(\widehat{\mathcal{L}} \cap \text{PSD})$ is trivial, e.g., when $A \succ 0$ and $b_l = b_u$ are finite, then we will have equality in Proposition 2.3. The proof makes use of an important external lemma but otherwise is self-contained. Related proof techniques have been used in [10, 18].

LEMMA 2.3 (Pataki [48]). *Consider a consistent feasibility system in the symmetric matrix variable Y , which enforces $Y \succeq 0$ as well as p linear equalities and q linear inequalities. Suppose \bar{Y} is an extreme point of the feasible set, and let $\bar{r} := \text{rank}(Y)$ and \bar{s} be the number of inactive linear inequalities at \bar{Y} . Then $\bar{r}(\bar{r}+1)/2 + \bar{s} \leq p + q$.*

THEOREM 2.7. *Let $\mathcal{F} = \{x \in \mathbb{R}^n : b_l \leq x^T Ax + a^T x \leq b_u\}$ with $-\infty \leq b_l \leq b_u \leq +\infty$ be nonempty. Then*

$$\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{L}} \cap \text{PSD} = \text{conv } \widehat{\mathcal{F}} + \text{rcone}(\widehat{\mathcal{L}} \cap \text{PSD}).$$

Proof. Proposition 2.3 gives the inclusion $\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{L}} \cap \text{PSD}$. So we need to prove $\widehat{\mathcal{L}} \cap \text{PSD} = \text{clconv } \widehat{\mathcal{F}} + \text{rcone}(\widehat{\mathcal{L}} \cap \text{PSD})$. The containment \supseteq is straightforward by construction.

For the containment \subseteq , recall that any point in a convex set may be written as a convex combination of finitely many extreme points plus a finite number of extreme rays. Hence, to complete the proof, it suffices to show that every extreme point of $\widehat{\mathcal{L}} \cap \text{PSD}$ is in $\widehat{\mathcal{F}}$.

So let (\bar{x}, \bar{X}) be any extreme point of $\widehat{\mathcal{L}} \cap \text{PSD}$. Examining (2.9) in the context of Lemma 2.3, $\widehat{\mathcal{L}} \cap \text{PSD}$ can be represented as a feasibility system in

$$Y := \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix}$$

under four scenarios based on the values of (b_l, b_u) :

- (i) $(p, q) = (1, 0)$ if both b_l and b_u are infinite;
- (ii) $(p, q) = (1, 1)$ if exactly one is finite;
- (iii) $(p, q) = (1, 2)$ if both are finite and $b_l < b_u$; or
- (iv) $(p, q) = (2, 0)$ if both are finite and $b_l = b_u$.

Define \bar{Y} according to (\bar{x}, \bar{X}) and (\bar{r}, \bar{s}) as in the lemma. In the three cases (i), (ii), and (iv), $p + q \leq 2$, and since $\bar{s} \geq 0$, we have

$$\bar{r}(\bar{r} + 1)/2 \leq \bar{r}(\bar{r} + 1)/2 + \bar{s} \leq p + q \leq 2,$$

in which case $\bar{r} \leq 1$. In the case (iii), $p + q = 3$ but $\bar{s} \geq 1$ since $b_l < b_u$, and so

$$\bar{r}(\bar{r} + 1)/2 \leq p + q - \bar{s} \leq 2,$$

which implies $\bar{r} \leq 1$ as well. Overall, $\bar{r} \leq 1$ means that (\bar{x}, \bar{X}) satisfies $\bar{X} = \bar{x}\bar{x}^T$, in which case $(\bar{x}, \bar{X}) \in \widehat{\mathcal{F}}$, as desired. \square

We remark that, if one were able to prove $\text{rcone}(\text{clconv } \widehat{\mathcal{F}}) = \text{rcone}(\widehat{\mathcal{L}} \cap \text{PSD})$, then one would have equality in general. Using Lemma 2.3, it is possible to show that $\text{rcone}(\widehat{\mathcal{L}} \cap \text{PSD}) = \widehat{\mathcal{D}}$, where

$$\begin{aligned} \mathcal{D} &:= \left\{ d \in \mathbb{R}^n : \begin{array}{ll} 0 \leq d^T A d & \text{if } -\infty < b_l \\ d^T A d \leq 0 & \text{if } b_u < +\infty \end{array} \right\} \\ \widehat{\mathcal{D}} &:= \{(0, dd^T) \in \mathbb{R}^n \times \mathcal{S}^n : d \in \mathcal{D}\}, \end{aligned}$$

but the relationship between $\widehat{\mathcal{D}}$ and $\text{rcone}(\text{clconv } \widehat{\mathcal{F}})$ is unclear.

3. Convex relaxations and valid inequalities: Related topics.

3.1. Another approach to convexification. We have presented Section 2 in terms of the set $\text{clconv } \widehat{\mathcal{F}}$. Another common approach [29, 58] is to study the so-called *convex* and *concave envelopes* of nonconvex functions. For example, in the formulation **(MIQCP)**, suppose $f(x) := x^T Cx + c^T x$ is nonconvex, and let $f_-(x)$ be any convex function that underestimates $f(x)$ in \mathcal{F} , i.e., $f_-(x) \leq f(x)$ holds for all $x \in \mathcal{F}$. Then one can relax $f(x)$ by $f_-(x)$. Considering all such $f_-(x)$, since the point-wise supremum of convex functions is convex, there is a single convex $\hat{f}_-(x)$ which most closely underestimates the objective. By definition, $\hat{f}_-(x)$ is the convex envelope for $f(x)$ over \mathcal{F} . The convex envelope is closely related to the closed convex hull of the graph or epigraph of $f(x)$, i.e., $\text{clconv}\{(x, f(x)) : x \in \mathcal{F}\}$ or $\text{clconv}\{(x, w) : x \in \mathcal{F}, f(x) \leq w\}$. Concave envelopes apply similar ideas to overestimation.

One can obtain a convex relaxation of **(MIQCP)** by relaxing the objective and all nonconvex constraints using convex envelopes. This can be

seen as an alternative to lifting via $X = xx^T$. Either approach is generically hard. Another alternative is to perform some mixture of lifting and convex envelopes as is done in [36].

Like the various results in Section 2 with $\text{clconv } \widehat{\mathcal{F}}$, there are relatively few cases (typically low-dimensional) for which exact convex envelopes are known. For example, the following gives another perspective on Theorem 2.2 and equation (2.3) above:

THEOREM 3.1 (Al-Khayyal and Falk [4]). *Let $\mathcal{F} = \{x \in \mathbb{R}^n : l \leq x \leq u\}$ with (l, u) finite in all components. For all $1 \leq i < j \leq n$, the convex and concave envelopes of $f(x_i, x_j) = x_i x_j$ over \mathcal{F} are, respectively,*

$$\begin{aligned} & \max\{l_i x_j + x_i l_j - l_i l_j, u_i x_j + x_i u_j - u_i u_j\} \\ & \min\{x_i u_j + l_i x_j - l_i u_j, x_i l_j + u_i x_j - u_i l_j\}. \end{aligned}$$

These basic formulas can be used to construct convex underestimators (not necessarily envelopes) for any quadratic function by separating that quadratic function into pieces based on all $x_i x_j$. Such techniques are used in the software BARON [51]. Also, [21] generalizes the above theorem to the case where f is the product of two linear functions having disjoint support.

3.2. A SOCP relaxation. [31] proposes an SOCP relaxation for **(MIQCP)**, which does not explicitly require the lifting $X = xx^T$ and is related to ideas in difference-of-convex programming [28].

First, the authors assume without loss of generality that the objective of **(MIQCP)** is linear. This can be achieved, for example, by introducing a new variable $t \in \mathbb{R}$ as well as a new quadratic constraint $x^T C x + c^T x \leq t$ and then minimizing t . Next, for each $k \in NQ$, A_k is written as the difference of two (carefully chosen) positive semidefinite $A_k^+, A_k^- \succeq 0$, i.e., $A_k = A_k^+ - A_k^-$, so that k -th constraint may be expressed as

$$x^T A_k^+ x + a_k^T x \leq b_k + x^T A_k^- x.$$

Then, an auxiliary variable $z_k \in \mathbb{R}$ is introduced to represent $x^T A_k^- x$ but also immediately relaxed as $x^T A_k^- x \leq z_k$ resulting in the convex system

$$\begin{aligned} & x^T A_k^+ x + a_k^T x \leq b_k + z_k \\ & x^T A_k^- x \leq z_k. \end{aligned}$$

Finally, z_k must be bounded in some fashion, say as $z_k \leq \mu_k \in \mathbb{R}$, or else the relaxation will in fact be useless. Bounding z_k depends very much on the problem and the choice of A_k^+, A_k^- . [31] provides strategies for bounding z_k .

The relaxation thus obtained can be modeled as a problem having only linear and convex quadratic inequalities, which can in turn be represented as an SOCP. In total, the relaxation obtained by the authors is

$$\left\{ \begin{array}{ll} a_k^T x \leq b_k & \forall k \in LQ \\ x^T A_k x + a_k^T x \leq b_k & \forall k \in CQ \\ x^T A_k^+ x + a_k^T x \leq b_k + z_k & \forall k \in NQ \\ x^T A_k^- x \leq z_k & \forall k \in NQ \\ l \leq x \leq u, z \leq \mu & \end{array} \right\}.$$

This SOCP model is shown to be dominated by the SDP relaxation $\widehat{\mathcal{L}} \cap \text{PSD}$, while it is not directly comparable to the basic LP relaxation $\widehat{\mathcal{L}}$.

The above relaxation was recently revisited in [53]. The authors studied the relaxation obtained by the following splitting of the A_k matrices,

$$A_k = \sum_{\lambda_{kj} > 0} \lambda_{kj} v_{kj} v_{kj}^T - \sum_{\lambda_{kj} < 0} \lambda_{kj} v_{kj} v_{kj}^T,$$

where $\{\lambda_{k1}, \dots, \lambda_{kn}\}$ and $\{v_{k1}, \dots, v_{kn}\}$ are the set of eigenvalues and eigenvectors of A_k , respectively. The constraint $x^T A_k x + a_k^T x \leq b_k$ can thus be reformulated as, $\sum_{\lambda_{kj} > 0} \lambda_{kj} (v_{kj}^T x)^2 + a_k^T x \leq b_k + \sum_{\lambda_{kj} < 0} \lambda_{kj} (v_{kj}^T x)^2$.

The non-convex terms $(v_{kj}^T x)^2$ ($\lambda_{kj} < 0$) can be relaxed by using their secant approximation to derive a convex relaxation of the above constraint. Instances of **(MIQCP)** tend to have geometric correlations along those v_{kj} with $\lambda_{kj} < 0$, which can be captured by projection techniques, and embedded within the polarity framework to derive strong cutting planes. We refer the reader to [53] for further details.

3.3. Results relating simple bounds and the binary integer grid. Motivated by Theorems such as 2.1 and 2.3 and the prior work of Padberg [44] and Yajima and Fujie [60], Burer and Letchford [19] studied the relationship between the two convex hulls

$$\text{clconv} \{(x, xx^T) : x \in [0, 1]^n\} \quad (3.1a)$$

$$\text{clconv} \{(x, xx^T) : x \in \{0, 1\}^n\}. \quad (3.1b)$$

The convex hull (3.1a) has been named QPB_n by the authors because of its relationship to “quadratic programming over the box.” The convex hull (3.1b) is essentially the well known boolean quadric polytope BQP_n [44]. In fact, the authors show that BQP_n is simply the coordinate projection of (3.1b) on the variables x_i ($1 \leq i \leq n$) and X_{ij} ($1 \leq i < j \leq n$). Note that nothing is lost in the projection because $X_{ii} = x_i$ and $X_{ji} = X_{ij}$ are valid for (3.1b).

We let π represent the coordinate projection just mentioned, i.e., onto the variables x_i and X_{ij} ($i < j$). The authors’ result can be stated as $\pi(QPB_n) = BQP_n$, which immediately implies the following:

THEOREM 3.2 (Burer and Letchford [19]). *Any inequality in the variables x_i ($1 \leq i \leq n$) and X_{ij} ($1 \leq i < j \leq n$), which is valid for BQP_n , is also valid for QPB_n .*

For proper interpretation of the theorem, it is important to keep in mind that QPB_n still involves the variables X_{ii} , while those same variables have been projected out to obtain BQP_n . So another way to phrase the theorem is as follows: a valid inequality for BQP_n becomes valid for QPB_n when the variables X_{ii} are introduced into the inequality with zero coefficients.

This result shows that, in a certain sense, describing QPB_n is at least as hard as describing BQP_n , and since many classes of valid inequalities are already known for BQP_n , it also gives many classes of valid inequalities for QPB_n . Indeed, the authors prove many classes of facets for BQP_n are in fact facets for QPB_n . The authors also demonstrate that PSD and the RLT inequalities for pairs (i, i) help describe QPB_n beyond BQP_n .

3.4. Completely positive programming. Burer [18] has recently studied a special case of **(MIQCP)** having

$$\mathcal{F} = \left\{ x \geq 0 : \begin{array}{l} Ax = b \\ x_i x_j = 0 \quad \forall (i, j) \in E \\ x_i \in \{0, 1\} \quad \forall i \in I \end{array} \right\}, \quad (3.2)$$

where, in particular, A is a rectangular matrix and $E \subseteq [n] \times [n]$ is symmetric. The author considers this specific form because it is amenable to analysis and yet is still fairly general. The results in [18] do not appear to hold, for example, under the general quadratic constraints of **(MIQCP)**.

Proposition 2.3 and similar logic as in Theorem 2.3 show that

$$\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{H}}_{\text{PSD}} := \widehat{\mathcal{L}} \cap \text{RLT} \cap R_2 \cap \text{PSD} \cap \{(x, X) : X_{ii} = x_i \quad \forall i \in I\}.$$

The following simplifying proposition is fairly easy to show:

PROPOSITION 3.1 (Burer [17]). *It holds that*

$$\widehat{\mathcal{H}}_{\text{PSD}} = \left\{ (x, X) \in \text{PSD} : \begin{array}{l} (x, X) \geq 0 \\ Ax = b, \text{diag}(AXA^T) = b^2 \\ X_{ij} = 0 \quad \forall (i, j) \in E \\ X_{ii} = x_i \quad \forall i \in I \end{array} \right\}.$$

Actually, $(x, X) \in \text{clconv } \widehat{\mathcal{F}}$ satisfies a stronger convex condition than $(x, X) \in \text{PSD}$. Recall that $(x, X) \in \text{PSD}$ is derived from

$$\begin{pmatrix} 1 & x^T \\ x & xx^T \end{pmatrix} = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^T \succeq 0.$$

Using that $x \in \mathcal{F}$ has $x \geq 0$, the above matrix is completely positive, not just positive semidefinite; see Section 2.5.3. We write

$$(x, X) \in \text{CPP} \iff \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \text{ is completely positive}$$

and define $\widehat{\mathcal{H}}_{\text{CPP}} := \widehat{\mathcal{H}}_{\text{PSD}} \cap \text{CPP}$. The result below establishes that $\text{clconv } \widehat{\mathcal{F}} = \widehat{\mathcal{H}}_{\text{CPP}}$.

THEOREM 3.3 (Burer [18], Bomze and Jarre [15]). *Let \mathcal{F} be defined as in (3.2). Define $J := \{j : \exists k \text{ s.t. } (j, k) \in E \text{ or } (k, j) \in E\}$, and suppose x_i is bounded in $\{x \geq 0 : Ax = b\}$ for all $i \in I \cup J$. Then $\text{clconv } \widehat{\mathcal{F}} = \widehat{\mathcal{H}}_{\text{CPP}}$.*

We emphasize that the result holds regardless of the boundedness of \mathcal{F} as a whole; it is only important that certain variables are bounded. Completely positive representations of $\text{clconv } \widehat{\mathcal{F}}$ for different \mathcal{F} , which are not already covered by the above theorem, can also be found in [49, 50].

Starting from the above theorem, Burer [17] has implemented a specialized algorithm for optimizing the relaxation $\widehat{\mathcal{H}}_{\text{PSD}}$. We briefly discuss this implementation in Section 5.

3.5. Higher-order liftings and projections. Whenever it is not possible to capture $\text{clconv } \widehat{\mathcal{F}}$ exactly in the lifted space (x, X) , it is still possible to lift into ever higher dimensional spaces and to linearize, say, cubic, quartic, or higher-degree valid inequalities there. This is quite a deep and powerful technique for capturing $\text{clconv } \widehat{\mathcal{F}}$. We refer the reader to the following papers: [9, 14, 33, 34, 35, 37, 54].

One of the most famous results in this area is the *sequential convexification* result for mixed 0-1 linear programs (M01LPs). Balas [8] showed that M01LPs are special cases of facial disjunctive programs which possess the *sequential convexifiability property*. Simply put, this means that the closed convex hull of all feasible solutions to a M01LP can be obtained by imposing the 0-1 condition on the binary variables *sequentially*, i.e., by imposing the 0-1 condition on the first binary variable and convexifying the resulting set, followed by imposing the 0-1 condition on the second variable, and so on. This is stated as the following theorem.

THEOREM 3.4 (Balas [8]). *Let \mathcal{F} be the feasible set of a M01LP, i.e.,*

$$\mathcal{F} = \{x \in \{0, 1\}^n : a_k^T x \leq b_k \forall k = 1, \dots, m\}$$

and define \mathcal{L} to be its basic linear relaxation in x . For each $i = 1, \dots, n$, define $T_i := \{x : x_i \in \{0, 1\}\}$. and

$$\begin{aligned} S_0 &:= \mathcal{L} \\ S_i &:= \text{clconv}(S_{i-1} \cap T_i) \quad \forall i = 1, \dots, n. \end{aligned}$$

Then $S_n = \text{clconv } \mathcal{F}$.

There exists an analogous sequential convexification for the continuous case of **(MIQCP)** for general quadratic constraints.

THEOREM 3.5 (Saxena et al. [52]). *Suppose that the feasible region \mathcal{F} of **(MIQCP)** is bounded with $I = \emptyset$, i.e., no integer variables. For each $i = 1, \dots, n$, define $\widehat{T}_i := \{(x, X) : X_{ii} \leq x_i^2\}$. Also define*

$$\begin{aligned} \widehat{S}_0 &:= \widehat{\mathcal{L}} \cap \text{PSD} \\ \widehat{S}_i &:= \text{clconv}(\widehat{S}_{i-1} \cap \widehat{T}_i) \quad \forall i = 1, \dots, n. \end{aligned}$$

Then $\widehat{S}_n = \text{clconv } \widehat{\mathcal{F}}$.

Part of the motivation for this theorem comes from the fact that

$$\text{PSD} \cap \bigcap_{i=1}^n \widehat{T}_i = \{(x, X) : X = xx^T\}$$

i.e., enforcing all \widehat{T}_i along with positive semidefiniteness recovers the non-convex condition $X = xx^T$. This is analogous to the fact that $\bigcap_{i=1}^n T_i$ recovers the integer condition in Theorem 3.4.

There is one crucial difference between Theorems 3.4 and 3.5. Note that a M01LP with a single binary variable is polynomial-time solvable; Balas [8] gave a polynomial-sized LP for this problem. On the other hand, the analogous problem in the context of **(MIQCP)** involves minimizing a linear function over a nonconvex set of the form

$$\widehat{\mathcal{L}} \cap \text{PSD} \cap \{(x, X) : X_{ii} \leq x_i^2\}.$$

It is not immediately clear if this is a polynomial-time solvable problem. Indeed, it is likely to be NP-hard [38].

An immediate consequence of any sequential convexification theorem is that it decomposes the non-convexity of the original problem (M01LP or MIQCP) into a set of simple *atomic* non-convex conditions, such as $x_j \in \{0, 1\}$ or $X_{ii} \leq x_i^2$ that can be handled separately. For instance, Balas, Ceria and Cornuéjols [9] studied a lifted LP formulation of M01LP with a single binary variable and combined it with projection techniques to derive a family of cutting planes for M01LP, widely known as lift-and-project cuts. In order to apply the same idea to **(MIQCP)** we need systematic techniques for deriving valid cutting planes for the set $\widehat{\mathcal{L}} \cap \text{PSD} \cap \{(x, X) : X_{ii} \leq x_i^2\}$; a disjunctive programming based approach is described in the following section.

Theorems 3.4 and 3.5 can actually be combined to convexify any bounded \mathcal{F} having a mix of binary and continuous variables. Also, Theorem 3.5 holds if the sets \widehat{T}_i are defined with respect to any orthonormal basis $\{v_1, \dots, v_n\}$, i.e., $\widehat{T}_i = \{(x, X) : \langle v_i v_i^T, X \rangle \leq (v_i^T x)^2\}$, not just the standard basis $\{e_1, \dots, e_n\}$. We refer the reader to [52] for proofs of these results.

4. Dynamic approaches for generating valid inequalities. Our starting point in this section is the lifted version $\widehat{\mathcal{F}}$ of the feasible set \mathcal{F} , whose convex hull can be relaxed, for example, as $\text{clconv } \widehat{\mathcal{F}} \subseteq \widehat{\mathcal{L}} \cap \text{RLT} \cap R_2 \cap S_2 \cap \text{PSD}$ (see Proposition 2.3). We are particularly interested in improving this relaxation through valid inequalities coming from a certain disjunctive programming approach.

Besides the presence of the integrality constraints $x_i \in \mathbb{Z}$, the only nonconvex constraint in $\widehat{\mathcal{F}}$ is the nonlinear equation $X = xx^T$ which can be represented exactly by the pair of SDP inequalities $X - xx^T \succeq 0$ and

$X - xx^T \preceq 0$. In fact, by the Schur complement theorem, the former is equivalent to the inequality (2.6), which is enforced by PSD. However, the latter is nonconvex. So relaxing $\widehat{\mathcal{F}}$ to $\widehat{\mathcal{L}} \cap \text{PSD}$ can be viewed as simply dropping $X - xx^T \preceq 0$. Said differently, $\widehat{\mathcal{F}} = \widehat{\mathcal{L}} \cap \text{PSD} \cap \{(x, X) : X - xx^T \preceq 0\}$. Harnessing the power of the inequality $X - xx^T \preceq 0$ constitutes the emphasis of this section.

As an aside, we would like to mention that all the results presented in this section exploit the continuous non-convexities in **(MIQCP)** to generate cutting planes. Non-convexities arising from presence of integer variables can be handled in a manner that is usually done in MILP; we refer the reader to [52] for computational results on disjunctive cuts for **(MIQCP)** derived from elementary 0-1 disjunctions, and to [7] for mixed integer rounding cuts for conic programs with 0-1 variables.

4.1. A procedure for generating disjunctive cuts. For any orthonormal basis $\{v_1, \dots, v_n\}$ of \mathbb{R}^n ,

$$\widehat{\mathcal{F}} = \widehat{\mathcal{L}} \cap \text{PSD} \cap \{(x, X) : X - xx^T \preceq 0\} \quad (4.1)$$

$$= \widehat{\mathcal{L}} \cap \text{PSD} \cap \{(x, X) : \langle X, v_i v_i^T \rangle \leq (v_i^T x)^2 \ \forall i = 1, \dots, n\}. \quad (4.2)$$

Given an arbitrary incumbent solution (\hat{x}, \hat{X}) , say, from optimizing over $\widehat{\mathcal{L}}$ or $\widehat{\mathcal{L}} \cap \text{PSD}$, we would like to choose a basis $\{v_1, \dots, v_n\}$ whose corresponding reformulation most effectively elucidates the infeasibility of (\hat{x}, \hat{X}) with respect to (4.1). The problem of choosing such a basis can be formulated as the following optimization problem that focuses on maximizing the violation of (\hat{x}, \hat{X}) with respect to the set of nonconvex constraints $\langle X, v_i v_i^T \rangle \leq (v_i^T x)^2$:

$$\begin{aligned} \max & \quad \max_{i=1 \dots n} \langle \hat{X}, v_i v_i^T \rangle - (v_i^T \hat{x})^2 \\ \text{s.t.} & \quad \{v_1, \dots, v_n\} \text{ is an orthonormal basis.} \end{aligned}$$

Clearly, a set of orthonormal eigenvectors of $\hat{X} - \hat{x}\hat{x}^T$ is an optimal solution to the above problem. This exercise of choosing a reformulation that hinges on certain characteristics of the incumbent solution (in this case the spectral decomposition of (\hat{x}, \hat{X})) can be viewed as a *dynamic* reformulation technique that rotates the coordinate axes so as to most effectively highlight the infeasibility of the incumbent solution.

Having chosen an orthonormal basis, we need a systematic technique to derive cutting planes for $\text{clconv } \widehat{\mathcal{F}}$ using (4.2). We use the framework of disjunctive programming to accomplish this goal. Classical disjunctive programming of Balas [8] requires a linear relaxation $\widehat{\mathcal{P}}$ of $\widehat{\mathcal{F}}$ and a disjunction that is satisfied by all $(x, X) \in \widehat{\mathcal{F}}$. The linear relaxation $\widehat{\mathcal{P}}$ could be taken equal to $\widehat{\mathcal{L}}$ but could also incorporate cutting planes generated from previous incumbent solutions. As for the choice of disjunctions, we seek the sources of nonconvexities in (4.2). Evidently, (4.2) has two of these,

namely, the integrality conditions on the variables x_i for $i \in I$ and the non-convex constraints $\langle X, v_i v_i^T \rangle \leq (v_i^T x)^2$. Integrality constraints have been used to derive disjunctions in MILP for the past five decades; examples of such disjunctions include elementary 0-1 disjunctions, split disjunctions, GUB disjunctions, etc. We do not detail these disjunctions here. For constraints of the type $\langle Y, vv^T \rangle \leq (v^T x)^2$ for fixed $v \in \mathbb{R}^n$, Saxena et. al. [52] proposed a technique to derive a valid disjunction, which we detail next. Following [52], we refer to $\langle X, vv^T \rangle \leq (v^T x)^2$ as a *univariate expression*.

Let

$$\begin{aligned}\eta_L(v) &:= \min \left\{ v^T x \mid (x, X) \in \widehat{\mathcal{P}} \right\} \\ \eta_U(v) &:= \max \left\{ v^T x \mid (x, X) \in \widehat{\mathcal{P}} \right\} \\ \theta &\in (\eta_L(v), \eta_U(v)).\end{aligned}$$

In their computational experiments, Saxena et. al. [52] chose $\theta = v^T \hat{x}$, where (\hat{x}, \hat{X}) is the current incumbent. Every $(x, X) \in \widehat{\mathcal{F}}$ satisfies the following disjunction:

$$\begin{aligned}& \left[\begin{array}{l} \eta_L(v) \leq v^T x \leq \theta \\ -(v^T x)(\eta_L(v) + \theta) + \theta \eta_L(v) \leq -\langle X, vv^T \rangle \end{array} \right] \vee \\ & \left[\begin{array}{l} \theta \leq v^T x \leq \eta_U(v) \\ -(v^T x)(\eta_U(v) + \theta) + \theta \eta_U(v) \leq -\langle X, vv^T \rangle \end{array} \right].\end{aligned}\tag{4.3}$$

This disjunction can be derived by splitting the range $[\eta_L(v), \eta_U(v)]$ of the function $v^T x$ over $\widehat{\mathcal{P}}$ into the two intervals $[\eta_L(v), \theta]$ and $[\theta, \eta_U(v)]$ and constructing a secant approximation of the function $-(v^T x)^2$ in each of the intervals, respectively (see Figure 1 for an illustration).

The disjunction (4.3) can then be embedded within the framework of Cut Generation Linear Programs (CGLPs) to derive disjunctive cuts as discussed in the following theorem.

THEOREM 4.1 ([8]). ¹Let a polyhedral set $P = \{x : Ax \geq b\}$, a disjunction $D = \bigvee_{k=1}^q (D_k x \geq d_k)$ and a point $\hat{x} \in P$ be given. Then $\hat{x} \in Q := \text{clconv} \cup_{k=1}^q \{x \in P \mid D_k x \geq d_k\}$ if and only if the optimal value of the following Cut Generation Linear Program (CGLP) is non-negative:

¹We caution the reader that the notation used in this theorem is not specifically tied to the notation for \mathcal{F} and related sets.

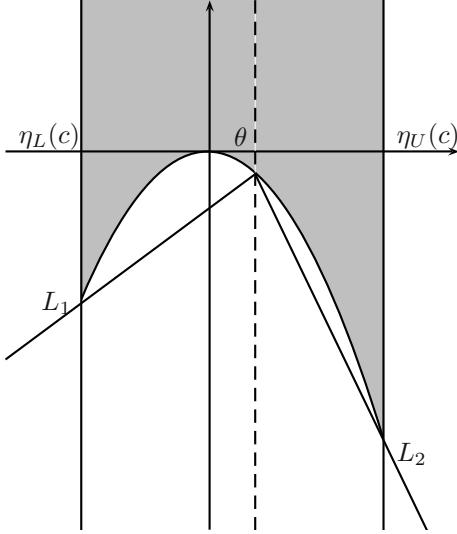


FIG. 1. The constraint $-(v^T x)^2 \leq -\langle X, vv^T \rangle$ and the disjunction (1) represented in the space spanned by $v^T x$ (horizontal axis) and $-\langle X, vv^T \rangle$ (vertical axis). The feasible region is the grey area above the parabola between $\eta_L(v)$ and $\eta_U(v)$. Disjunction (4.3) is obtained by taking the piecewise-linear approximation of the parabola, using a breakpoint at θ , and given by the two lines L_1 and L_2 . Clearly, if $\eta_L(v) \leq v^T x \leq \theta$ then (x, X) must be above L_1 to be in the grey area; if $\theta \leq v^T x \leq \eta_U(v)$ then (x, X) must be above L_2 .

$$\begin{aligned}
 \min \quad & \alpha^T \hat{x} - \beta && (\text{CGLP}) \\
 \text{s.t.} \quad & A^T u^k + D_k^T v^k = \alpha \quad k = 1, \dots, q \\
 & b^T u^k + d_k^T v^k \geq \beta \quad k = 1, \dots, q \\
 & u^k, v^k \geq 0 \quad k = 1, \dots, q \\
 & \sum_{k=1}^q (\xi^T u^k + \xi_k^T v^k) = 1
 \end{aligned}$$

where ξ and ξ_k ($k = 1, \dots, q$) are any non-negative vectors of conformable dimensions that satisfy $\xi_k > 0$ ($k = 1, \dots, q$). If the optimal value of (CGLP) is negative, and (α, β) are part of an optimal solution, then $\alpha^T x \geq \beta$ is a valid inequality for Q which cuts off \hat{x} .

Next, we illustrate the above procedure for deriving disjunctive cuts on a small example. Consider the following instance of **(MIQCP)** derived from the st_ph11 instance from the GLOBALLib repository [27]:

$$\begin{aligned}
 \min \quad & x_1 + x_2 + x_3 - \frac{1}{2} (x_1^2 + x_2^2 + x_3^2) \\
 \text{s.t.} \quad & 2x_1 + 3x_2 + 4x_3 \leq 35 \\
 & 0 \leq x_1, x_2, x_3 \leq 4.
 \end{aligned}$$

An optimal solution to the linear-semidefinite relaxation

$$\min \left\{ x_1 + x_2 + x_3 - \frac{1}{2}(X_{11} + X_{22} + X_{33}) : (x, X) \in \hat{\mathcal{L}} \cap \text{PSD} \right\}$$

is

$$\hat{x} = \begin{pmatrix} 4 \\ 4 \\ 3.75 \end{pmatrix}, \quad \hat{X} = \begin{pmatrix} 16 & 16 & 15 \\ 16 & 16 & 15 \\ 15 & 15 & 15 \end{pmatrix}$$

and so

$$\hat{X} - \hat{x}\hat{x}^T = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.9375 \end{pmatrix}$$

has exactly one non-zero eigenvalue. The associated eigenvector and univariate expression are given by $c^T = (0, 0, 1)$ and $X_{33} \leq x_3^2$, respectively. Note that (\hat{x}, \hat{X}) satisfies the secant approximation $X_{33} \leq 4x_3$ of $X_{33} \leq x_3^2$ at equality; hence *the secant inequality does not cut off this point*. Choosing $\theta = 2$ in (4.3), we get the following disjunction which is satisfied by every feasible solution $(x, X) \in \hat{\mathcal{F}}$ for this example:

$$\left[\begin{array}{l} 0 \leq x_3 \leq 2 \\ 2x_3 - X_{33} \geq 0 \end{array} \right] \vee \left[\begin{array}{l} 2 \leq x_3 \leq 4 \\ 6x_3 - X_{33} \geq 8 \end{array} \right].$$

In order to derive a disjunctive cut, for each term in the disjunction we sum a non-negative weighted combination of its constraints together with the original linear constraints of $\hat{\mathcal{F}}$ to construct a new constraint valid for that term in the disjunction. If the separate weights in each term can be chosen in such a way that the resulting constraints for both terms are the same, then that constraint is a disjunctive cut. In particular, using the weighting scheme

$$\left[\begin{array}{ll} 2x_3 - y_{33} \geq 0 & (14.70588) \\ -x_1 \geq -4 & (15.68627) \\ -x_2 \geq -4 & (23.52941) \\ x_3 \geq 0 & (27.45098) \end{array} \right] \vee \left[\begin{array}{ll} 6x_3 - y_{33} \geq 8 & (14.70588) \\ -2x_1 - 3x_2 - 4x_3 \geq -35 & (7.84314) \end{array} \right],$$

we arrive at the disjunctive cut

$$-15.68627x_1 - 23.52941x_2 + 56.86275x_3 - 14.70588y_{33} \geq -156.86274.$$

This disjunctive cut is violated by (\hat{x}, \hat{X}) .

This example highlights a very important aspect of disjunctive programming: its ability to involve additional problem constraints in deriving strong cuts for $\text{clconv } \widehat{\mathcal{F}}$. For illustration, consider the same example and the relaxation $\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$. Note that this relaxation only incorporates the effect of the general linear constraint $2x_1 + 3x_2 + 4x_3 \leq 35$ via the set $\widehat{\mathcal{L}}$. Defining

$$x^1 = \begin{pmatrix} 4 \\ 4 \\ 4 \end{pmatrix}, \quad X^1 = x^1(x^1)^T \quad x^2 = \begin{pmatrix} 4 \\ 4 \\ 0 \end{pmatrix}, \quad X^2 = x^2(x^2)^T$$

and $(\hat{x}, \hat{X}) := \frac{15}{16}(x^1, X^1) + \frac{1}{16}(x^2, X^2)$, i.e., the same (\hat{x}, \hat{X}) is in the example, it holds that $(\hat{x}, \hat{X}) \in \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$. Note, however, that the endpoint (x^1, X^1) is not in $\text{clconv } \widehat{\mathcal{F}}$ since $x^1 \notin \mathcal{F}$, which implies $(x^1, X^1) \notin \widehat{\mathcal{L}}$ in this case. So it remains possible that (\hat{x}, \hat{X}) is not in $\text{clconv } \widehat{\mathcal{F}}$. Indeed, by explicitly involving the linear constraint in a more powerful way during the convexification process, disjunctive programming cuts off (\hat{x}, \hat{X}) from $\text{clconv } \widehat{\mathcal{F}}$.

In fact, something stronger can be said. For this same example, define $\mathcal{F}_{lu} := \{x : 0 \leq x_1, x_2, x_3 \leq 4\}$ so that $\mathcal{F} = \mathcal{F}_{lu} \cap \{x : 2x_1 + 3x_2 + 4x_3 \leq 35\}$; also define $\widehat{\mathcal{F}}_{lu}$ accordingly. Now consider the stronger relaxation $\widehat{\mathcal{L}} \cap \text{clconv } \widehat{\mathcal{F}}_{lu}$ of $\text{clconv } \widehat{\mathcal{F}}$, which completely convexifies with respect to the bounds $l \leq x \leq u$. Still, even this stronger relaxation contains (\hat{x}, \hat{X}) , and so we see that convexifying with respect to the bounds is simply not enough to cut off (\hat{x}, \hat{X}) . One must incorporate the general linear inequality in a more aggressive fashion such as disjunctive programming does.

4.2. Computational insights. In [52], the authors report computational results with a cutting plane procedure based on these ideas. For instances of **(MIQCP)** coming from GLOBALLib, the authors solved five separate relaxations of

$$v_* := \min \left\{ \langle C, X \rangle + c^T x : x \in \text{clconv } \widehat{\mathcal{F}} \right\}.$$

These relaxations were (with accompanying “version numbers” and optimal values for reference)

$$\begin{aligned} (\text{V0}) \quad v_{\text{RLT}} &:= \min \quad \langle C, X \rangle + c^T x \\ &\text{s.t.} \quad x \in \widehat{\mathcal{L}} \cap \text{RLT}, \end{aligned}$$

$$\begin{aligned} (\text{V1}) \quad v_{\text{PSD}} &:= \min \quad \langle C, X \rangle + c^T x \\ &\text{s.t.} \quad x \in \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}, \end{aligned}$$

$$\begin{aligned}
(V2) \quad v_{dsj} &:= \min \quad \langle C, X \rangle + c^T x \\
&\text{s.t.} \quad x \in \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD} \cap \text{“disjunctive cuts”}, \\
(V2-SI) \quad v_{sec} &:= \min \quad \langle C, X \rangle + c^T x \\
&\text{s.t.} \quad x \in \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD} \cap \text{“secant cuts”}, \\
(V2-Dsj) \quad v'_{dsj} &:= \min \quad \langle C, X \rangle + c^T x \\
&\text{s.t.} \quad x \in \widehat{\mathcal{L}} \cap \text{RLT} \cap \text{“disjunctive cuts”}.
\end{aligned}$$

The secant-cut referred to in the description of V2-SI is obtained by constructing the convex envelope of the non-convex inequality $\langle Y, vv^T \rangle \leq (v^T x)^2$; using the notation introduced above, the corresponding secant inequality is given by, $\langle Y, vv^T \rangle \leq (\eta_L(v) + \eta_U(v))v^T x - \eta_L(v)\eta_U(v)$. Since the secant inequality can be obtained cheaply once $\eta_L(v)$ and $\eta_U(v)$ have been computed, variant V2-SI helps us assess the marginal importance of using the computationally expensive disjunctive cut as compared to readily available secant inequality.

Note that $v_* \leq v_{dsj} \leq v_{sec} \leq v_{PSD} \leq v_{RLT}$ and $v_* \leq v_{dsj} \leq v'_{dsj} \leq v_{RLT}$. V0 was used as a base relaxation by which others were judged. In particular, for each of the four remaining relaxations, the metric

$$\text{percent duality gap closed} := \frac{v_{RLT} - v}{v_{RLT} - v_*} \times 100$$

was recorded on each instance using the optimal value v for that relaxation. Only instances having $v_{RLT} > v_*$ were selected for testing (129 instances). We remark that, when present, constraint PSD was enforced with a cutting-plane approach based on convex quadratic cuts rather than a black-box SDP solver.

TABLE 1
Summary of computational results.

	V1	V2	V2-SI	V2-Dsj
>99.99 %	16	23	24	1
98-99.99 %	1	44	4	29
75-98 %	10	23	17	10
25-75 %	11	22	26	29
0-25 %	91	17	58	60
Average Gap Closed	24.80%	76.49%	44.40%	41.54%

Table 1 summarizes the authors' key results on the 129 instances. Each of the main columns gives, for that version, the number of instances in several bins of the metric “percentage gap closed.” Some comments are in order.

First, the variant V2 code that uses disjunctive cuts closes 50% more duality gap than the SDP relaxation V1. In fact, relaxations obtained by adding disjunctive cuts close more than 98% of the duality gap on 67 out of 129 instances; the same figure for SDP relaxations is 17 out of 129 instances. Second, the authors were able to close 99% of the duality gap on some of the instances such as st_qpc-m3a, st_ph13, st_ph11, ex3_1_4, st_jcbpaf2, ex2_1_9, etc., on which the SDP relaxation closes 0% of the duality gap.

Third, the variant V2-SI of the code that uses the secant inequality instead of disjunctive cuts does close a significant proportion (44.40%) of the duality gap. However, using disjunctive cuts improves this statistic to 76.49% thereby demonstrating the marginal benefits of disjunctive programming. Fourth, it is worth observing that both variants V2 and V2-SI have access to the same kinds of nonconvexities, namely, univariate expressions $\langle X, vv^T \rangle \leq (v^T x)^2$ derived from eigenvectors v of $\hat{X} - \hat{x}\hat{x}^T$. Despite this commonality, why does V2, which has access to the CGLP apparatus, outperform V2-SI? The answer to this question lies in the way the individual frameworks process the nonconvex expression $\langle X, vv^T \rangle \leq (v^T x)^2$. While V2-SI takes a local view of the problem and convexifies $\langle X, vv^T \rangle \leq (v^T x)^2$ in the 2-dimensional space spanned by $v^T x$ and $\langle X, vv^T \rangle$, V2 takes a global view of the problem and combines disjunctive terms with other problem constraints. It is precisely this ability to derive stronger inferences by combining disjunctive information with other problem constraints that allows V2 to outperform its local counterpart V2-SI.

Fifth, it is worth observing that removing PSD has a debilitating effect on the cutting plane algorithm presented in [52] as demonstrated by the performance of V2-Dsj relative to V2. While the CGLP apparatus allows us to take a global view of the problem, its ability to derive strong disjunctive cuts is limited by the strength of the initial relaxation. By removing PSD, the relaxation is significantly weakened, and this subsequently has a deteriorating effect on the strength of disjunctive cuts later derived.

TABLE 2
Selection criteria.

% Duality Gap closed by		Instance Chosen
V1	V2	
< 10 %	> 90 %	st_jcbpaf2
> 40%	< 60%	ex9_2_7
< 10%	< 10%	ex7_3_1

The basic premise of the work in [52] lies in generating valid cutting planes for $\text{clconv } \widehat{\mathcal{F}}$ from the spectrum of $\hat{X} - \hat{x}\hat{x}^T$, where (\hat{x}, \hat{X}) is the incumbent solution. In order to highlight the impact of these cuts on

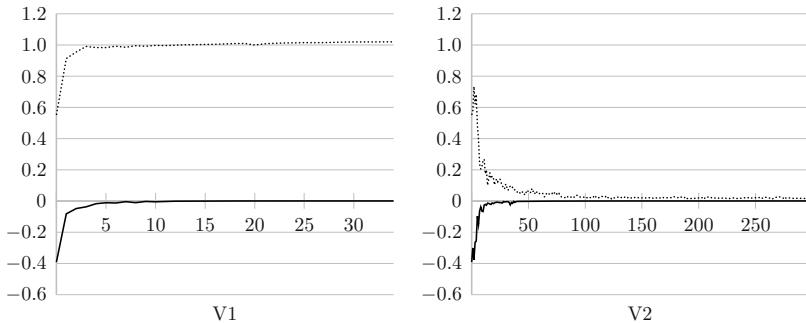


FIG. 2. Plot of the sum of positive and negative eigenvalues for *st-jcbaf2* with $V1-V2$.

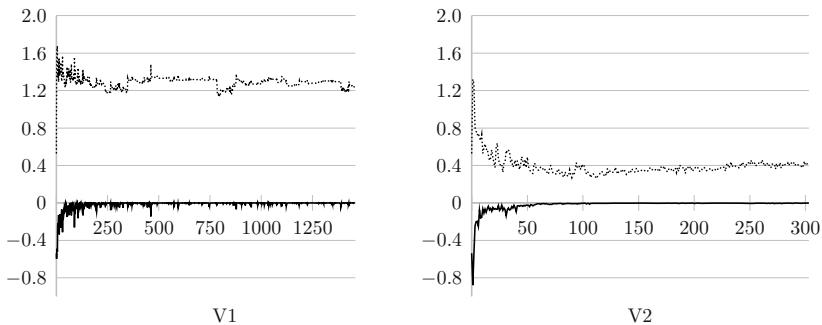


FIG. 3. Plot of the sum of positive and negative eigenvalues for *ex_9_2_7* with $V1-V2$.

the spectrum itself, the authors presented details on three instances listed in Table 2 which we reproduce here for the sake of illustration. Figures 2–4 report the key results. The horizontal axis represents the number of iterations while the vertical axis reports the sum of the positive eigenvalues of $\hat{X} - \hat{x}\hat{x}^T$ (broken line) and the sum of the negative eigenvalues of $\hat{X} - \hat{x}\hat{x}^T$ (solid line). Some remarks are in order.

First, the graph of the sum of negative eigenvalues converges to zero much faster than the corresponding graph for positive eigenvalues. This is not surprising because the problem of eliminating the negative eigenvalues is a convex programming problem, namely an SDP; the approach of adding convex-quadratic cuts is just an iterative cutting-plane based technique to impose the $X - xx^T \succeq 0$ condition. Second, V1 has a widely varying effect on the sum of positive eigenvalues of $X - xx^T$. This is to be expected because the $X - xx^T \succeq 0$ condition imposes no constraint on the positive eigenvalues of $X - xx^T$. Furthermore, the sum of positive eigenvalues represents the part of the nonconvexity of $\hat{\mathcal{F}}$ that is not captured by PSD. Third, it is interesting to note that the variant that uses disjunctive cuts,

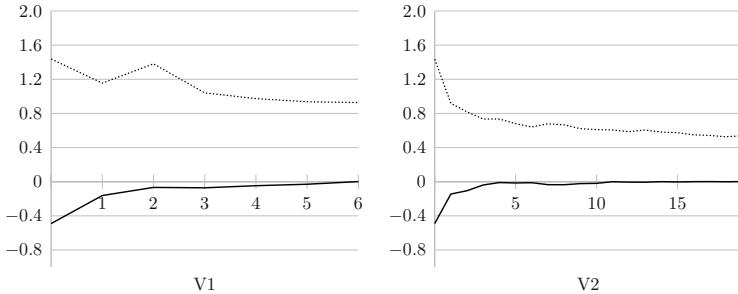


FIG. 4. Plot of the sum of the positive and negative eigenvalues for ex_7-3-1 with V1–V2.

namely V2, is able to force both positive and negative eigenvalues to converge to 0 for the st_jcbpaf2 thereby generating an almost feasible solution to this problem.

4.3. Working with only the original variables. Finally, we would like to emphasize that all of the relaxations of $\text{clconv } \widehat{\mathcal{F}}$ discussed until now are defined in the lifted space of (x, X) . While the additional variable X enhances the expressive power of the formulation, it also increases the size of the formulation drastically, resulting in an enormous computational overhead which would be, for example, incurred at every node of a branch-and-bound tree. Ideally, we would like to extract the strength of these extended reformulations in the form of cutting planes that are defined only in the space of the original x variable. Systematic approaches for constructing such convex relaxations of **(MIQCP)** are described in a recent paper by Saxena et. al. [53]. We briefly reproduce some of these results to expose the reader to this line of research.

Consider the relaxation $\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$ of $\text{clconv } \widehat{\mathcal{F}}$, and define $\mathcal{Q} := \text{proj}_x(\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD})$, which is a relaxation of $\text{clconv } \mathcal{F}$ (not $\text{clconv } \widehat{\mathcal{F}}$) in the space of the original variable x —but one that retains the power of $\widehat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}$. Can we separate from \mathcal{Q} , hence enabling us to work solely in the x -space? Specifically, given a point \hat{x} that satisfies at least the simple bounds $l \leq x \leq u$, we desire an algorithmic framework that either shows $\hat{x} \in \mathcal{Q}$ or finds an inequality valid for \mathcal{Q} which cuts off \hat{x} . Note that $\hat{x} \in \mathcal{Q}$ if and only if the following system is feasible in X with \hat{x} fixed:

$$\begin{aligned} \langle A_k, X \rangle + a_k^T \hat{x} &\leq b_k \quad \forall k = 1, \dots, m \\ l\hat{x}^T + \hat{x}l^T - ll^T \\ u\hat{x}^T + \hat{x}u^T - uu^T \end{aligned} \left\{ \right. \leq X \leq \hat{x}u^T + l\hat{x}^T - lu^T$$

$$\begin{pmatrix} 1 & \hat{x}^T \\ \hat{x} & X \end{pmatrix} \succeq 0.$$

As is typical, if this system is infeasible, then duality theory provides a cut (in this case, a convex quadratic cut) cutting off \hat{x} from \mathcal{Q} . Further, one

can optimize to obtain a deep cut. We refer the reader to [53] for further details where the authors report computational results to demonstrate the computational dividends of working in the space of original variables possibly augmented by a few additional variables. We reproduce a slice of their computational results in Section 5.

5. Computational case study. To give the reader an impression of the computational requirements of the relaxations and techniques proposed in this paper, we compare three implementations for solving the relaxation

$$\min\{\langle C, X \rangle + c^T x : (x, X) \in \hat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD}\} \quad (5.1)$$

of the following particular case of **(MIQCP)**, which is called *quadratic programming over the box*:

$$\min\{x^T C x + c^T x : x \in [0, 1]^n\}. \quad (5.2)$$

We compare a black-box interior-point-method SDP solver (called IPM for “interior-point method”), the specialized completely-positive solver of [17] mentioned in Section 3.4 (called CP for “completely positive”), and the projection cutting plane method of [53] discussed in Section 4.3 (called PROJ for “projection”). We refer the reader to the original papers for full details of the implementations.

Methods IPM and CP work with the formulation (5.1). On the other hand, PROJ first reformulates (5.2) as

$$\min \left\{ t : \begin{array}{l} x \in [0, 1]^n \\ x^T Q x + c^T x \leq t \end{array} \right\} \quad (5.3)$$

and then, in a pre-processing step, calculates several convex quadratic constraints as cutting planes for the relaxation $\text{proj}_{(t,x)}(\hat{\mathcal{L}} \cap \text{RLT} \cap \text{PSD})$ of the reformulation. The procedure for calculating the cutting planes is outlined briefly in Section 4.3. Theoretically, if all possible cuts are generated, then the power of (5.1) is recovered. In practice, however, it is hoped that a few deep cuts will recover most of the power of (5.1) but save a significant amount of computation time. Finally, letting $\alpha_k t^2 + \beta_k t + x^T A_k x + a_k^T x \leq b_k$ represent the derived convex cuts, PROJ solves the relaxation

$$\min \left\{ t : \begin{array}{l} x \in [0, 1]^n \\ \alpha_k t^2 + \beta_k t + x^T A_k x + a_k^T x \leq b_k \quad \forall k \end{array} \right\}. \quad (5.4)$$

Nine instances from [20] are tested. Their relevant characteristics under relaxations (5.1) and (5.4) are given in [Table 3](#), and the timings (in seconds) are given in [Table 4](#). Also, in [Table 5](#), we give the percentage gaps closed by the three methods relative to the pure linear relaxation $\hat{\mathcal{L}} \cap \text{RLT}$ (see Section 4.2 for a careful definition of the percentage gap closed). A few comments are in order.

TABLE 3
Sizes of tested instances.

Instance	# Variables		Linear		# Constraints	
	IPM/CP	PROJ	IPM/CP	PROJ	IPM/CP (SDP)	PROJ (quad)
spar100-025-1	5151	203	20201	156	1	119
spar100-025-2	5151	201	20201	151	1	95
spar100-025-3	5151	201	20201	150	1	114
spar100-050-1	5151	201	20201	150	1	98
spar100-050-2	5151	201	20201	150	1	113
spar100-050-3	5151	201	20201	150	1	97
spar100-075-1	5151	201	20201	150	1	131
spar100-075-2	5151	201	20201	150	1	109
spar100-075-3	5151	199	20201	147	1	90

TABLE 4
Computational utility of projected relaxations.

Instances	Time (sec)		
	IPM	CP	PROJ (pre-process + solve)
spar100-025-1	5719.42	59	670.15 + 1.14
spar100-025-2	10185.65	54	538.03 + 1.52
spar100-025-3	5407.09	58	656.59 + 1.24
spar100-050-1	10139.57	76	757.14 + 1.07
spar100-050-2	5355.20	92	929.91 + 1.26
spar100-050-3	7281.26	76	747.46 + 0.82
spar100-075-1	9660.79	101	1509.96 + 2.00
spar100-075-2	6576.10	100	936.61 + 1.23
spar100-075-3	10295.88	81	657.84 + 0.87

First, on each instance, IPM is not competitive with either CP or PROJ. This illustrates a recognized trend in solving relaxations of this sort, namely that, at this point in time, specialized solvers perform better than black-box ones. Perhaps this will change as black-box solvers become more robust. Second, CP performs best in terms of overall time on each instance, but PROJ, discounting its pre-processing phase, solves its relaxation the quickest while still closing most of the gap that IPM and CP do. Within the context of using PROJ within branch-and-bound, this accrues significance due to two observations: (i) most contemporary branch-and-bound procedures generate cutting planes primarily at the root node and only sparingly at other nodes; and (ii) such a relaxation would be solved hundreds or thousands of times within the tree. So the pre-processing time of PROJ can be effectively amortized over the entire branch-and-bound tree.

TABLE 5
Computational utility of projected relaxations.

Instances	% Gap Closed	
	IPM/CP	PROJ
spar100-025-1	98.93%	92.36%
spar100-025-2	99.09%	92.16%
spar100-025-3	99.33%	93.26%
spar100-050-1	98.17%	93.62%
spar100-050-2	98.57%	94.13%
spar100-050-3	99.39%	95.81%
spar100-075-1	99.19%	95.84%
spar100-075-2	99.18%	96.47%
spar100-075-3	99.19%	96.06%

We also mention that, while PROJ is currently being solved by a non-linear programming algorithm, the convex quadratic constraints of PROJ could actually be approximated by polyhedral relaxations introduced by Ben-Tal and Nemirovski [12] (also see [59]) yielding LP relaxations of these problems. Such LP relaxations are extremely desirable for branch-and-bound algorithms for two reasons. One, they can be efficiently re-optimized using warm-starting capabilities of LP solvers thereby reducing the computational overhead at nodes of the enumeration tree. Two, these LP relaxations can easily avail techniques, such as branching strategies, cutting planes, heuristics, etc., which have been developed by the MILP community in the past five decades (see [1] for application of these techniques in the context of convex MINLPs).

6. Conclusion. Table 6 catalogues the results covered in this paper. The first column lists the main concepts while the following two columns list their manifestations for M01LP and MIQCP, respectively. Some remarks are in order.

First, while linear programming based relaxations are almost universally used in M01LP, the same does not hold for MIQCP. There is a wide variety of relaxations for MIQCP that can be used, starting from the extended RLT+SDP relaxations to the compact eigen-reformulations (see [53]) defined in the original space of variables. It must be noted that all of these relaxations are currently solved by interior point methods that lack efficient re-optimization capabilities making them bottlenecks in a branch-and-bound procedure.

Second, there is a well established theory of exact formulations in M01LP (see [22]). Many of these results were obtained as byproducts of the tremendous amount of research that went into proving the perfect graph conjecture. Unfortunately, the progress in this direction in MIQCP

TABLE 6
M01LP vs. MIQCP.

Concept	M01LP	MIQCP
Relaxation	LP relaxation	$\hat{\mathcal{L}} \cap \text{RLT} \cap R_2 \cap S_2 \cap \text{PSD}$ projected SDP eigen-reformulation
Exact Description	total unimodularity; perfect, ideal, and balanced matrices	theorems in Section 2.5
Elementary Non-Convexity	$x_j \in \{0, 1\}$	$X_{ii} \leq x_i^2$
Linear Transformed Non-Convexity	$(\pi x \leq \pi_0) \vee$ $(\pi x \geq \pi_0 + 1)$	$\langle X, vv^T \rangle \leq (v^T x)^2$
Sequential Convexification	Balas [8]	Saxena et al. [52]

has been rather slow, and exact descriptions are unknown for most classes of problems except for some very small problem instances.

Third, there is an interesting connection between cuts derived from the univariate expression $\langle X, vv^T \rangle \leq (v^T x)^2$ for **MIQCP** and split cuts derived from split disjunctions $(\pi x \leq \pi_0) \vee (\pi x \geq \pi_0 + 1)$ ($\pi \in \mathbb{Z}^n$) in M01LP. To see this, note that $\langle X, vv^T \rangle \leq (v^T x)^2$ can be obtained from the elementary non-convex constraint $X_{ii} \leq x_i^2$ by the linear transformation $(x, X) \rightarrow (v^T x, \langle X, vv^T \rangle)$ where the linear transformation is chosen depending on the incumbent solution; for example, Saxena et al. [52] derive the v vector from the spectral decomposition of $\hat{X} - \hat{x}\hat{x}^T$. Similarly, the split disjunction $(\pi x \leq \pi_0) \vee (\pi x \geq \pi_0 + 1)$ can be obtained from elementary 0-1 disjunction $(x_j \leq 0) \vee (x_j \geq 1)$ by the linear transformation $x \rightarrow \pi x$ where the linear transformation is chosen depending on the incumbent solution; for instance, the well known mixed integer Gomory cuts can be obtained from split disjunctions derived by monoidal strengthening of elementary 0-1 disjunctions, wherein the monoid that is chosen to strengthen the cut depends on the incumbent solution (see [9]).

Acknowledgements. The authors are in debt to two anonymous referees for many helpful suggestions that have improved the paper significantly.

REFERENCES

- [1] K. ABHISHEK, S. LEYFFER, AND J.T. LINDEROTH, *FilmINT: An outer-approximation-based solver for convex mixed-integer nonlinear programs*, INFORMS Journal on Computing, **22** (2010), pp. 555–567.

- [2] T.K. ACHTERBERG, T. BERTHOLD AND K.T. WOLTER, *Constraint integer programming: A new approach to integrate cp and mip*, Lecture Notes in Computer Science, **5015** (2008), pp. 6–20.
- [3] F.A. AL-KHAYYAL, *Generalized bilinear programming, Part i: Models, applications, and linear programming relaxations*, European Journal of Operational Research, **60** (1992), pp. 306–314.
- [4] F.A. AL-KHAYYAL AND J.E. FALK, *Jointly constrained biconvex programming*, Math. Oper. Res., **8** (1983), pp. 273–286.
- [5] F. ALIZADEH AND D. GOLDFARB, *Second-order cone programming*, Math. Program., **95** (2003), pp. 3–51. ISMP 2000, Part 3 (Atlanta, GA).
- [6] K.M. ANSTREICHER AND S. BURER, *Computable representations for convex hulls of low-dimensional quadratic forms*, with K. Anstreicher, Mathematical Programming (Series B), **124**(1-2), pp. 33–43 (2010).
- [7] A. ATAMTÜRK AND V. NARAYANAN, *Conic mixed-integer rounding cuts*, Math. Program., **122** (2010), pp. 1–20.
- [8] E. BALAS, *Disjunctive programming: properties of the convex hull of feasible points*, Discrete Appl. Math., **89** (1998), pp. 3–44.
- [9] E. BALAS, S. CERIA, AND G. CORNUÉJOLS, *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, Mathematical Programming, **58** (1993), pp. 295–324.
- [10] A. BECK, *Quadratic matrix programming*, SIAM J. Optim., **17** (2006), pp. 1224–1238 (electronic).
- [11] P. BELOTTI, *Disjunctive cuts for non-convex MINLP*, IMA Volume Series, Springer 2010, accepted.
<http://myweb.clemson.edu/~pbelotti/papers/belotti-disj-MINLP.pdf>.
- [12] A. BEN-TAL AND A. NEMIROVSKI, *On polyhedral approximations of the second-order cone*, Math. Oper. Res., **26** (2001), pp. 193–205.
- [13] A. BERMAN AND N. SHAKED-MONDERER, *Completely Positive Matrices*, World Scientific, 2003.
- [14] D. BIENSTOCK AND M. ZUCKERBERG, *Subset algebra lift operators for 0-1 integer programming*, SIAM J. Optim., **15** (2004), pp. 63–95 (electronic).
- [15] I. BOMZE AND F. JARRE, *A note on Burer's copositive representation of mixed-binary QPs*, Optimization Letters, **4** (2010), pp. 465–472.
- [16] P. BONAMI, L. BIEGLER, A. CONN, G. CORNUÉJOLS, I. GROSSMANN, C. LAIRD, J. LEE, A. LODI, F. MARGOT, N. SAWAYA, AND A. WÄCHTER, *An algorithmic framework for convex mixed-integer nonlinear programs*, Discrete Optimization, **5** (2008), pp. 186–204.
- [17] S. BURER, *Optimizing a polyhedral-semidefinite relaxation of completely positive programs*, Mathematical Programming Computation, **2**(1), pp 1–19 (2010).
- [18] ———, *On the copositive representation of binary and continuous nonconvex quadratic programs*, Mathematical Programming, **120** (2009), pp. 479–495.
- [19] S. BURER AND A.N. LETCHFORD, *On nonconvex quadratic programming with box constraints*, SIAM J. Optim., **20** (2009), pp. 1073–1089.
- [20] S. BURER AND D. VANDENBUSSCHE, *Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound*, Comput. Optim. Appl., **43** (2009), pp. 181–195.
- [21] D. COPPERSMITH, O. GÜNLÜK, J. LEE, AND J. LEUNG, *A polytope for a product of a real linear functions in 0/1 variables*, manuscript, IBM, Yorktown Heights, NY, December 2003.
- [22] G. CORNUÉJOLS, *Combinatorial optimization: packing and covering*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.
- [23] R.W. COTTLE, G.J. HABETLER, AND C.E. LEMKE, *Quadratic forms semi-definite over convex cones*, in Proceedings of the Princeton Symposium on Mathematical Programming (Princeton Univ., 1967), Princeton, N.J., 1970, Princeton Univ. Press, pp. 551–565.

- [24] G. DANNINGER AND I.M. BOMZE, *Using copositivity for global optimality criteria in concave quadratic programming problems*, Math. Programming, **62** (1993), pp. 575–580.
- [25] G. DANTZIG, R. FULKERSON, AND S. JOHNSON, *Solution of a large-scale traveling salesman problem*, J. Operations Res. Soc. Amer., **2** (1954), pp. 393–410.
- [26] E. DE KLERK AND D.V. PASECHNIK, *Approximation of the stability number of a graph via copositive programming*, SIAM J. Optim., **12** (2002), pp. 875–892.
- [27] See the website: www.gamsworld.org/global/globallib/globalstat.htm.
- [28] R. HORST AND N.V. THOAI, *DC programming: overview*, J. Optim. Theory Appl., **103** (1999), pp. 1–43.
- [29] M. JACH, D. MICHAELS, AND R. WEISMANTEL, *The convex envelope of $(N - 1)$ -convex functions*, SIAM J. Optim., **19** (2008), pp. 1451–1466.
- [30] R. JEROSLOW, *There cannot be any algorithm for integer programming with quadratic constraints*, Operations Research, **21** (1973), pp. 221–224.
- [31] S. KIM AND M. KOJIMA, *Second order cone programming relaxation of non-convex quadratic optimization problems*, Optim. Methods Softw., **15** (2001), pp. 201–224.
- [32] ———, *Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations*, Comput. Optim. Appl., **26** (2003), pp. 143–154.
- [33] M. KOJIMA AND L. TUNÇEL, *Cones of matrices and successive convex relaxations of nonconvex sets*, SIAM J. Optim., **10** (2000), pp. 750–778.
- [34] J.B. LASSEUR, *Global optimization with polynomials and the problem of moments*, SIAM J. Optim., **11** (2001), pp. 796–817.
- [35] M. LAURENT, *A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming*, Math. Oper. Res., **28** (2003), pp. 470–496.
- [36] J. LINDEROTH, *A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs*, Math. Program., **103** (2005), pp. 251–282.
- [37] L. LOVÁSZ AND A. SCHRIJVER, *Cones of matrices and set-functions and 0-1 optimization*, SIAM Journal on Optimization, **1** (1991), pp. 166–190.
- [38] T. MATSUI, *NP-hardness of linear multiplicative programming and related problems*, J. Global Optim., **9** (1996), pp. 113–119.
- [39] J.E. MAXFIELD AND H. MINC, *On the matrix equation $X'X = A$* , Proc. Edinburgh Math. Soc. (2), **13** (1962/1963), pp. 125–129.
- [40] G.P. MCCORMICK, *Computability of global solutions to factorable nonconvex programs. I. Convex underestimating problems*, Math. Programming, **10** (1976), pp. 147–175.
- [41] See the website: <http://www.gamsworld.org/minlp/>.
- [42] K.G. MURTY AND S.N. KABADI, *Some NP-complete problems in quadratic and nonlinear programming*, Math. Programming, **39** (1987), pp. 117–129.
- [43] G. NEMHAUSER AND L. WOLSEY, *Integer and Combinatorial Optimization*, Wiley-Interscience, 1999.
- [44] M. PADBERG, *The Boolean quadric polytope: some characteristics, facets and relatives*, Math. Programming, **45** (1989), pp. 139–172.
- [45] P. PARDALOS, *Global optimization algorithms for linearly constrained indefinite quadratic problems*, Computers and Mathematics with Applications, **21** (1991), pp. 87–97.
- [46] P.M. PARDALOS AND S.A. VAVASIS, *Quadratic programming with one negative eigenvalue is NP-hard*, J. Global Optim., **1** (1991), pp. 15–22.
- [47] P. PARRILLO, *Structured Semidefinite Programs and Semi-algebraic Geometry Methods in Robustness and Optimization*, PhD thesis, California Institute of Technology, 2000.
- [48] G. PATAKI, *On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues*, Mathematics of Operations Research, **23** (1998), pp. 339–358.
- [49] J. POVH AND F. RENDL, *A copositive programming approach to graph partitioning*, SIAM J. Optim., **18** (2007), pp. 223–241.

- [50] ———, *Copositive and semidefinite relaxations of the quadratic assignment problem*, Discrete Optim., **6** (2009), pp. 231–241.
- [51] N.V. SAHINIDIS, *BARON: a general purpose global optimization software package*, J. Glob. Optim., **8** (1996), pp. 201–205.
- [52] A. SAXENA, P. BONAMI, AND J. LEE, *Convex relaxations of non-convex mixed integer quadratically constrained programs: Extended formulations*, Mathematical Programming (Series B), **124**(1-2), pp. 383–411 (2010). <http://dx.doi.org/10.1007/s10107-010-0371-9>.
- [53] ———, *Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations*, 2010. To appear in *Mathematical Programming*. <http://dx.doi.org/10.1007/s10107-010-0340-3>.
- [54] H.D. SHERALI AND W.P. ADAMS, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM J. Discrete Math., **3** (1990), pp. 411–430.
- [55] H.D. SHERALI AND W.P. ADAMS, *A Reformulation-Linearization Technique (RLT) for Solving Discrete and Continuous Nonconvex Problems*, Kluwer, 1997.
- [56] N. SHOR, *Quadratic optimization problems*, Soviet Journal of Computer and Systems Science, **25** (1987), pp. 1–11. Originally published in *Tekhnicheskaya Kibernetika*, **1**:128–139, 1987.
- [57] J.F. STURM AND S. ZHANG, *On cones of nonnegative quadratic functions*, Math. Oper. Res., **28** (2003), pp. 246–267.
- [58] M. TAWARMALANI AND N.V. SAHINIDIS, *Convexification and global optimization in continuous and mixed-integer nonlinear programming*, Vol. **65** of Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, 2002. Theory, algorithms, software, and applications.
- [59] J.P. VIELMA, S. AHMED, AND G.L. NEMHAUSER, *A lifted linear programming branch-and-bound algorithm for mixed-integer conic quadratic programs*, INFORMS J. Comput., **20** (2008), pp. 438–450.
- [60] Y. YAJIMA AND T. FUJIE, *A polyhedral approach for nonconvex quadratic programming problems with box constraints*, J. Global Optim., **13** (1998), pp. 151–170.
- [61] Y. YE AND S. ZHANG, *New results on quadratic minimization*, SIAM J. Optim., **14** (2003), pp. 245–267 (electronic).

LINEAR PROGRAMMING RELAXATIONS OF QUADRATICALLY CONSTRAINED QUADRATIC PROGRAMS

ANDREA QUALIZZA*, PIETRO BELOTTI†, AND FRANÇOIS MARGOT*‡

Abstract. We investigate the use of linear programming tools for solving semidefinite programming relaxations of quadratically constrained quadratic problems. Classes of valid linear inequalities are presented, including sparse *PSD* cuts, and principal minors *PSD* cuts. Computational results based on instances from the literature are presented.

Key words. Quadratic programming, semidefinite programming, relaxation, linear programming.

AMS(MOS) subject classifications. 90C57.

1. Introduction. Many combinatorial problems have Linear Programming (*LP*) relaxations that are commonly used for their solution through branch-and-cut algorithms. Some of them also have stronger relaxations involving positive semidefinite (*PSD*) constraints. In general, stronger relaxations should be preferred when solving a problem, thus using these *PSD* relaxations is tempting. However, they come with the drawback of requiring a Semidefinite Programming (*SDP*) solver, creating practical difficulties for an efficient implementation within a branch-and-cut algorithm. Indeed, a major weakness of current *SDP* solvers compared to *LP* solvers is their lack of efficient warm starting mechanisms. Another weakness is solving problems involving a mix of *PSD* constraints and a large number of linear inequalities, as these linear inequalities put a heavy toll on the linear algebra steps required during the solution process.

In this paper, we investigate *LP* relaxations of *PSD* constraints with the aim of capturing most of the strength of the *PSD* relaxation, while still being able to use an *LP* solver. The *LP* relaxation we obtain is an outer-approximation of the *PSD* cone, with the typical convergence difficulties when aiming to solve problems to optimality. We thus do not cast this work as an efficient way to solve *PSD* problems, but we aim at finding practical ways to approximate *PSD* constraints with linear ones.

We restrict our experiments to Quadratically Constrained Quadratic Programs (*QCQP*). A *QCQP* problem with variables $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ is a problem of the form

*Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213.

†Department of Mathematical Sciences, Clemson University, Clemson, SC 29634.

‡Corresponding author (fmargin@andrew.cmu.edu). Supported by NSF grant NSF-0750826.

$$\begin{aligned}
\max \quad & x^T Q_0 x + a_0^T x + b_0^T y \\
\text{s.t.} \quad & x^T Q_k x + a_k^T x + b_k^T y \leq c_k \quad \text{for } k = 1, 2, \dots, p \\
& l_{x_i} \leq x_i \leq u_{x_i} \quad \text{for } i = 1, 2, \dots, n \\
& l_{y_j} \leq y_j \leq u_{y_j} \quad \text{for } j = 1, 2, \dots, m
\end{aligned} \tag{QCQP}$$

where, for $k = 0, 1, 2, \dots, p$, Q_k is a rational symmetric $n \times n$ -matrix, a_k is a rational n -vector, b_k is a rational m -vector, and $c_k \in \mathbb{Q}$. Moreover, the lower and upper bounds l_{x_i}, u_{x_i} for $i = 1, \dots, n$, and l_{y_j}, u_{y_j} for $j = 1, \dots, m$ are all finite. If Q_0 is negative semidefinite and Q_k is positive semidefinite for each $k = 1, 2, \dots, p$, problem **QCQP** is convex and thus easy to solve. Otherwise, the problem is NP-hard [6].

An alternative *lifted* formulation for **QCQP** is obtained by replacing each quadratic term $x_i x_j$ with a new variable X_{ij} . Let $X = xx^T$ be the matrix with entry X_{ij} corresponding to the quadratic term $x_i x_j$. For square matrices A and B of the same dimension, let $A \bullet B$ denote the *Frobenius inner product* of A and B , i.e., the trace of $A^T B$. Problem **QCQP** is then equivalent to

$$\begin{aligned}
\max \quad & Q_0 \bullet X + a_0^T x + b_0^T y \\
\text{s.t.} \quad & Q_k \bullet X + a_k^T x + b_k^T y \leq c_k \quad \text{for } k = 1, 2, \dots, p \\
& l_{x_i} \leq x_i \leq u_{x_i} \quad \text{for } i = 1, 2, \dots, n \\
& l_{y_j} \leq y_j \leq u_{y_j} \quad \text{for } j = 1, 2, \dots, m \\
& X = xx^T.
\end{aligned} \tag{LIFT}$$

The difficulty in solving problem **LIFT** lies in the non-convex constraint $X = xx^T$. A relaxation, dubbed *PSD*, that is possible to solve relatively efficiently is obtained by relaxing this constraint to the requirement that $X - xx^T$ be positive semidefinite, i.e., $X - xx^T \succeq 0$. An alternative relaxation of **QCQP**, dubbed *RLT*, is obtained by the Reformulation Linearization Technique [17], using products of pairs of original constraints and bounds and replacing nonlinear terms with new variables.

Anstreicher [2] compares the *PSD* and *RLT* relaxations on a set of quadratic problems with box constraints, i.e., **QCQP** problems with $p = 0$ and with all the variables bounded between 0 and 1. He shows that the *PSD* relaxations of these instances are fairly good and that combining the *PSD* and *RLT* relaxations yields significantly tighter relaxations than either of the *PSD* or *RLT* relaxations. The drawback of combining the two relaxations is that current SDP solvers have difficulties to handle the large number of linear constraints of the *RLT*.

Our aim is to solve relaxations of **QCQP** using exclusively linear programming tools. The *RLT* is readily applicable for our purposes, while the *PSD* technique requires a cutting plane approach as described in Section 2.

In Section 3 we consider several families of valid cuts. The focus is essentially on capturing the strength of the positive semidefinite condition using standard cuts [18], and some sparse versions of these.

We analyze empirically the strength of the considered cuts on instances taken from GLOBALLib [10] and quadratic programs with box constraints described in more details in the next section. Implementation and computational results are presented in Section 4. Finally, Section 5 summarizes the results and gives possible directions for future research.

2. Relaxations of QCQP problems. A typical approach to get bounds on the optimal value of a **QCQP** is to solve a convex relaxation. Since our aim is to work with linear relaxations, the first step is to linearize **LIFT** by relaxing the last constraint to $X = X^T$. We thus get the Extended formulation

$$\begin{aligned} \max \quad & Q_0 \bullet X + a_0^T x + b_0^T y \\ \text{s.t.} \quad & Q_k \bullet X + a_k^T x + b_k^T y \leq c_k \quad \text{for } k = 1, 2, \dots, p \\ & l_{x_i} \leq x_i \leq u_{x_i} \quad \text{for } i = 1, 2, \dots, n \\ & l_{y_j} \leq y_j \leq u_{y_j} \quad \text{for } j = 1, 2, \dots, m \\ & X = X^T. \end{aligned} \tag{\textbf{EXT}}$$

EXT is a Linear Program with $n(n+3)/2 + m$ variables and the same number of constraints as **QCQP**. Note that the optimal value of **EXT** is usually a weak upper bound for **QCQP**, as no constraint links the values of the x and X variables. Two main approaches for doing that have been proposed and are based on relaxations of the last constraint of **LIFT**, namely

$$X - xx^T = 0. \tag{2.1}$$

They are known as the Positive Semidefinite (*PSD*) relaxation and the Reformulation Linearization Technique (*RLT*) relaxation.

2.1. PSD relaxation. As $X - xx^T = 0$ implies $X - xx^T \succcurlyeq 0$, using this last constraint yields a convex relaxation of **QCQP**. This is the approach used in [18, 20, 21, 23], among others.

Moreover, using Schur's complement

$$X - xx^T \succcurlyeq 0 \iff \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succcurlyeq 0,$$

and defining

$$\tilde{Q}_k = \begin{pmatrix} -c_k & a_k^T/2 \\ a_k/2 & Q_k \end{pmatrix}, \quad \tilde{X} = \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix},$$

we can write the *PSD* relaxation of **QCQP** in the compact form

$$\begin{aligned} \max \quad & \tilde{Q}_0 \bullet \tilde{X} + b_0^T y \\ \text{s.t.} \quad & \begin{aligned} \tilde{Q} \bullet \tilde{X} + b_k^T y \leq 0 & \quad k = 1, 2, \dots, p \\ l_{x_i} \leq x_i \leq u_{x_i} & \quad i = 1, 2, \dots, n \\ l_{y_j} \leq y_j \leq u_{y_j} & \quad j = 1, 2, \dots, m \\ \tilde{X} \succcurlyeq 0. & \end{aligned} \end{aligned} \tag{\textbf{PSD}}$$

This is a positive semidefinite problem with linear constraints. It can thus be solved in polynomial time using interior point algorithms. **PSD** is tighter than usual linear relaxations for problems such as the Maximum Cut, Stable Set, and Quadratic Assignment problems [25]. All these problems can be formulated as **QCQPs**.

2.2. RLT relaxation. The Reformulation Linearization Technique [17] can be used to produce a relaxation of **QCQP**. It adds linear inequalities to **EXT**. These inequalities are derived from the variable bounds and constraints of the original problem as follows: multiply together two original constraints or bounds and replace each product term $x_i x_j$ with the variable X_{ij} . For instance, let $x_i, x_j, i, j \in \{1, 2, \dots, n\}$ be two variables from **QCQP**. By taking into account only the four original bounds $x_i - l_{x_i} \geq 0$, $x_i - u_{x_i} \leq 0$, $x_j - l_{x_j} \geq 0$, $x_j - u_{x_j} \leq 0$, we get the *RLT* inequalities

$$\begin{aligned} X_{ij} - l_{x_i} x_j - l_{x_j} x_i & \geq -l_{x_i} l_{x_j}, \\ X_{ij} - u_{x_i} x_j - u_{x_j} x_i & \geq -u_{x_i} u_{x_j}, \\ X_{ij} - l_{x_i} x_j - u_{x_j} x_i & \leq -l_{x_i} u_{x_j}, \\ X_{ij} - u_{x_i} x_j - l_{x_j} x_i & \leq -u_{x_i} l_{x_j}. \end{aligned} \tag{2.2}$$

Anstreicher [2] observes that, for Quadratic Programs with box constraints, the *PSD* and *RLT* constraints together yield much better bounds than those obtained from the **PSD** or **RLT** relaxations. In this work, we want to capture the strength of both techniques and generate a Linear Programming relaxation of **QCQP**.

Notice that the four inequalities above, introduced by McCormick [12], constitute the convex envelope of the set $\{(x_i, x_j, X_{ij}) \in \mathbb{R}^3 : l_{x_i} \leq x_i \leq u_{x_i}, l_{x_j} \leq x_j \leq u_{x_j}, X_{ij} = x_i x_j\}$ as proven by Al-Khayyal and Falk [1], i.e., they are the tightest relaxation for the single term X_{ij} .

3. Our framework. While the *RLT* constraints are linear in the variables in the **EXT** formulation and therefore can be added directly to **EXT**, this is not the case for the *PSD* constraint. We use a linear outer-approximation of the **PSD** relaxation and a cutting plane framework, adding a linear inequality separating the current solution from the *PSD* cone.

The initial relaxation we use and the various cuts generated by our separation procedure are described in more details in the next sections.

3.1. Initial relaxation. Our initial relaxation is the **EXT** formulation together with the $O(n^2)$ **RLT** constraints derived from the bounds on the variables x_i , $i = 1, 2, \dots, n$. We did not include the **RLT** constraints derived from the problem constraints due to their large number and the fact that we want to avoid the introduction of extra variables for the multivariate terms that occur when quadratic constraints are multiplied together.

The bounds $[L_{ij}, U_{ij}]$ for the extended variables X_{ij} are computed as follows:

$$\begin{aligned} L_{ij} &= \min\{l_{x_i} l_{x_j}; l_{x_i} u_{x_j}; u_{x_i} l_{x_j}; u_{x_i} u_{x_j}\}, \quad \forall i = 1, 2, \dots, n; j = i, \dots, n \\ U_{ij} &= \max\{l_{x_i} l_{x_j}; l_{x_i} u_{x_j}; u_{x_i} l_{x_j}; u_{x_i} u_{x_j}\}, \quad \forall i = 1, 2, \dots, n; j = i, \dots, n. \end{aligned}$$

In addition, equality (2.1) implies $X_{ii} \geq x_i^2$. We therefore also make sure that $L_{ii} \geq 0$. In the remainder of the paper, this initial relaxation is identified as **EXT+RLT**.

3.2. PSD cuts. We use the equivalence that a matrix is positive semidefinite if and only if

$$v^T \tilde{X} v \geq 0 \quad \text{for all } v \in \mathbb{R}^{n+1}. \quad (3.1)$$

We can reformulate **PSD** as the semi-infinite Linear Program

$$\begin{aligned} \max \quad & \tilde{Q}_0 \bullet \tilde{X} + b_0^T y \\ \text{s.t.} \quad & \begin{aligned} \tilde{Q} \bullet \tilde{X} + b_k^T y &\leq c_k && \text{for } k = 1, 2, \dots, p \\ l_{x_i} \leq x_i \leq u_{x_i} &&& \text{for } i = 1, 2, \dots, n \\ l_{y_j} \leq y_j \leq u_{y_j} &&& \text{for } j = 1, 2, \dots, m \\ v^T \tilde{X} v &\geq 0 && \text{for all } v \in \mathbb{R}^{n+1}. \end{aligned} \end{aligned} \quad (\text{PSDLP})$$

A practical way to use **PSDLP** is to adopt a cutting plane approach to separate constraints (3.1) as done in [18].

Let \tilde{X}^* be an arbitrary point in the space of the \tilde{X} variables. The spectral decomposition of \tilde{X}^* is used to decide if \tilde{X}^* is in the **PSD** cone or not. Let the eigenvalues and corresponding orthonormal eigenvectors of \tilde{X}^* be λ_k and v_k for $k = 1, 2, \dots, n$, and assume without loss of generality that $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ and let $t \in \{0, \dots, n\}$ such that $\lambda_t < 0 \leq \lambda_{t+1}$. If $t = 0$, then all the eigenvalues are non negative and \tilde{X}^* is positive semidefinite. Otherwise, $v_k^T \tilde{X}^* v_k = v_k^T \lambda_k v_k = \lambda_k < 0$ for $k = 1, \dots, t$. Hence, the valid cut

$$v_k^T \tilde{X} v_k \geq 0 \quad (3.2)$$

is violated by \tilde{X}^* . Cuts of the form (3.2) are called **PSDCUTs** in the remainder of the paper.

The above procedure has two major weaknesses: First, only one cut is obtained from eigenvector v_k for $k = 1, \dots, t$, while computing the spectral

```

Sparsify( $v, \tilde{X}, pct_{NZ}, pct_{VIOL}$ )
1    $min_{VIOL} \leftarrow -v^T \tilde{X}v \cdot pct_{VIOL}$ 
2    $max_{NZ} \leftarrow \lfloor length[v] \cdot pct_{NZ} \rfloor$ 
3    $w \leftarrow v$ 
4    $perm \leftarrow$  random permutation of 1 to  $length[w]$ 
5   for  $j \leftarrow 1$  to  $length[w]$ 
6     do
7        $z \leftarrow w, z[perm[j]] \leftarrow 0$ 
8       if  $-z^T \tilde{X}z > min_{VIOL}$ 
9         then  $w \leftarrow z$ 
10  if number of non-zeroes in  $w < max_{NZ}$ 
11    then output  $w$ 

```

FIG. 1. *Sparsification procedure for PSD cuts.*

decomposition requires a non trivial investment in cpu time, and second, the cuts are usually very dense, i.e. almost all entries in vv^T are nonzero. Dense cuts are frowned upon when used in a cutting plane approach, as they might slow down considerably the reoptimization of the linear relaxation.

To address these weaknesses, we describe in the next section a heuristic to generate several sparser cuts from each of the vectors v_k for $k = 1, \dots, t$.

3.3. Sparsification of PSD cuts. A simple idea to get sparse cuts is to start with vector $w = v_k$, for $k = 1, \dots, t$, and iteratively set to zero some component of w , provided that $w^T \tilde{X}^* w$ remains sufficiently negative. If the entries are considered in random order, several cuts can be obtained from a single eigenvector v_k . For example, consider the *Sparsify* procedure in Figure 1, taking as parameters an initial vector v , a matrix \tilde{X} , and two numbers between 0 and 1, pct_{NZ} and pct_{VIOL} , that control the maximum percentage of nonzero entries in the final vector and the minimum violation requested for the corresponding cut, respectively. In the procedure, parameter $length[v]$ identifies the size of vector v .

It is possible to implement this procedure to run in $O(n^2)$ if $length[v] = n + 1$: Compute and update a vector m such that

$$m_j = \sum_{i=1}^{n+1} w_j w_i \tilde{X}_{ij} \text{ for } j = 1, \dots, n + 1 .$$

Its initial computation takes $O(n^2)$ and its update, after a single entry of w is set to 0, takes $O(n)$. The vector m can be used to compute the left hand side of the test in step 8 in constant time given the value of the violation d for the inequality generated by the current vector w : Setting the entry $\ell = perm[j]$ of w to zero reduces the violation by $2m_\ell - w_\ell^2 \tilde{X}_{\ell\ell}$ and thus the violation of the resulting vector is $(d - 2m_\ell + w_\ell^2 \tilde{X}_{\ell\ell})$.

A slight modification of the procedure is used to obtain several cuts from the same eigenvector: Change the loop condition in step 5 to consider the entries in $perm$ in cyclical order, from all possible starting points s in $\{1, 2, \dots, length[w]\}$, with the additional condition that entry $s - 1$ is not set to 0 when starting from s to guarantee that we do not generate always the same cut. From our experiments, this simple idea produces collections of sparse and well-diversified cuts. This is referred to as SPARSE1 in the remainder of the paper.

We also consider the following variant of the procedure given in Figure 1. Given a vector w , let $\tilde{X}_{[w]}$ be the principal minor of \tilde{X} induced by the indices of the nonzero entries in w . Replace step 7 with

7. $z \leftarrow \bar{w}$ where \bar{w} is an eigenvector corresponding to the most negative eigenvalue of a spectral decomposition of $\tilde{X}_{[w]}$, $z[perm[j]] \leftarrow 0$.

This is referred to as SPARSE2 in the remainder, and we call the cuts generated by SPARSE1 or SPARSE2 described above *Sparse PSD cuts*.

Once sparse *PSD* cuts are generated, for each vector w generated, we can also add all *PSD* cuts given by the eigenvectors corresponding to negative eigenvalues of a spectral decomposition of $\tilde{X}_{[w]}$. These cuts are valid and sparse. They are called *Minor PSD cuts* and denoted by MINOR in the following.

An experiment to determine good values for the parameters pct_{NZ} and pct_{VIOL} was performed on the 38 GLOBALLIB instances and 51 BoxQP instances described in Section 4.1. It is run by selecting two sets of three values in $[0, 1]$, $\{V_{LOW}, V_{MID}, V_{UP}\}$ for pct_{VIOL} and $\{N_{LOW}, N_{MID}, N_{UP}\}$ for pct_{NZ} . The nine possible combinations of these parameter values are used and the best of the nine (V_{best}, N_{best}) is selected. We then center and reduce the possible ranges around V_{best} and N_{best} , respectively, and repeat the operation. The procedure is stopped when the best candidate parameters are (V_{MID}, N_{MID}) and the size of the ranges satisfy $|V_{UP} - V_{LOW}| \leq 0.2$ and $|N_{UP} - N_{LOW}| \leq 0.1$.

In order to select the best value of the parameters, we compare the bounds obtained by both algorithms after 1, 2, 5, 10, 20, and 30 seconds of computation. At each of these times, we count the number of times each algorithm outperforms the other by at least 1% and the winner is the algorithm with the largest number of wins over the 6 clocked times. It is worth noting that typically the majority of the comparisons end up as ties, implying that the results are not extremely sensitive to the selected values for the parameters.

For SPARSE1, the best parameter values are $pct_{VIOL} = 0.6$ and $pct_{NZ} = 0.2$. For SPARSE2, they are $pct_{VIOL} = 0.6$ and $pct_{NZ} = 0.4$. These values are used in all experiments using either SPARSE1 or SPARSE2 in the remainder of the paper.

4. Computational results. In the implementation, we have used the Open Solver Interface (Osi-0.97.1) from COIN-OR [8] to create and modify the LPs and to interface with the LP solvers ILOG Cplex-11.1. To compute eigenvalues and eigenvectors, we use the `dSYEVX` function provided by the LAPACK library version 3.1.1. We also include a cut management procedure to reduce the number of constraints in the outer approximation LP. This procedure, applied at the end of each iteration, removes the cuts that are not satisfied with equality by the optimal solution. Note however that the constraints from the **EXT+RLT** formulation are never removed, only constraints from added cutting planes are possibly removed.

The machine used for the tests is a 64 bit 2.66GHz AMD processor, 64GB of RAM memory, and Linux kernel 2.6.29. Tolerances on the accuracy of the primal and dual solutions of the LP solver and LAPACK calls are set to 10^{-8} .

The set of instances used for most experiments consists of 51 BoxQP instances with at most 50 variables and the 38 GLOBALLib instances as described in Section 4.1.

For an instance \mathcal{I} and a given relaxation of it, we define the *gap closed* by the relaxation as

$$100 \cdot \frac{RLT - BND}{RLT - OPT}, \quad (4.1)$$

where BND and RLT are the optimal value for the given relaxation and the **EXT+RLT** relaxation respectively, and OPT is either the optimal value of \mathcal{I} or the best known value for a feasible solution. The OPT values are taken from [14].

4.1. Instances. Tests are performed on a subset of instances from GLOBALLib [10] and on Box Constrained Quadratic Programs (BoxQPs) [24]. GLOBALLib contains 413 continuous global optimization problems of various sizes and types, such as BoxQPs, problems with complementarity constraints, and general QCQPs. Following [14], we select 160 instances from GLOBALLib having at most 50 variables and that can easily be formulated as **QCQP**. The conversion of a non-linear expression into a quadratic expression, when possible, is performed by adding new variables and constraints to the problem. Additionally, bounds on the variables are derived using linear programming techniques and these bound are included in the formulation. From these 160 instances in AMPL format, we substitute each bilinear term $x_i x_j$ by the new variable X_{ij} as described for the **LIFT** formulation. We build two collections of linearized instances in MPS format, one with the original precision on the coefficients and right hand side, and the second with 8-digit precision. In our experiments we used the latter.

As observed in [14], using together the **SDP** and **RLT** relaxations yields stronger bounds than those given by the **RLT** relaxation only for 38

out of 160 GLOBALLib instances. Hence, we focus on these 38 instances to test the effectiveness of the *PSD* Cuts and their sparse versions.

The BoxQP collection contains 90 instances with a number of variables ranging from 20 to 100. Due to time limit constraints and the number of experiments to run, we consider only instances with a number of variables between 20 to 50, for a total of 51 BoxQP problems.

The converted GLOBALLib and BoxQP instances are available in MPS format from [13].

4.2. Effectiveness of each class of cuts. We first compare the effectiveness of the various classes of cuts when used in combination with the standard PSDCUTs. For these tests, at most 1,000 cutting iterations are performed, at most 600 seconds are used, and operations are stopped if tailing off is detected. More precisely, let z_t be the optimal value of the linear relaxation at iteration t . The operations are halted if $t \geq 50$ and $z_t \geq (1 - 0.0001) \cdot z_{t-50}$. A cut purging procedure is used to remove cuts that are not tight at iteration t if the condition $z_t \geq (1 - 0.0001) \cdot z_{t-1}$ is satisfied. On average in each iteration the algorithm generates $\frac{n^2}{2}$ cuts, of which only $\frac{n}{2}$ are kept by the cut purging procedure and the rest are discarded.

In order to compare two different cutting plane algorithms, we compare the closed gap values first after a fixed number of iterations, and second at several given times, for all QCQP instances at avail. Comparisons at fixed iterations indicate the quality of the cuts, irrespective of the time used to generate them. Comparisons at given times are useful if only limited time is available for running the cutting plane algorithms and a good approximation of the *PSD* cone is sought. The closed gaps obtained at a given point are deemed different only if their difference is at least $g\%$ of the initial gap. We report comparisons for $g = 1$ and $g = 5$. Comparisons at one point is possible only if both algorithms reach that point. The number of problems for which this does not happen – because, at a given time, either result was not available or one of the two algorithms had already stopped, or because either algorithm had terminated in fewer iterations – is listed in the “inc.” (incomparable) columns in the tables. For the remaining problems, we report the percentage of problems for which one algorithm is better than the other and the percentage of problems were they are tied. Finally, we also report the average improvement in gap closed for the second algorithm over the first algorithm in the column labeled “impr.”.

Tests are first performed to decide which combination of the SPARSE1, SPARSE2 and MINOR cuts perform best on average. Based on Tables 1 and 2 below, we conclude that using MINOR is useful both in terms of iteration and time, and that the algorithm using PSDCUT+SPARSE2+MINOR (abbreviated *S2M* in the remainder) dominates the algorithm using PSDCUT+SPARSE1+MINOR (abbreviated *S1M*) both in terms of iteration and time. Table 1 gives the comparison between S1M and S2M at differ-

TABLE 1
Comparison of S1M with S2M at several iterations.

Iteration	g = 1			g = 5			inc.	impr.
	S1M	S2M	Tie	S1M	S2M	Tie		
1	7.87	39.33	52.80	1.12	19.1	79.78	0.00	3.21
2	17.98	28.09	53.93	0.00	10.11	89.89	0.00	2.05
3	17.98	19.10	62.92	1.12	7.87	91.01	0.00	1.50
5	12.36	14.61	73.03	3.37	5.62	91.01	0.00	1.77
10	10.11	13.48	76.41	0.00	5.62	94.38	0.00	1.42
15	4.49	13.48	82.03	1.12	6.74	92.14	0.00	1.12
20	1.12	10.11	78.66	1.12	6.74	82.02	10.11	1.02
30	1.12	8.99	79.78	1.12	5.62	83.15	10.11	0.79
50	2.25	6.74	80.90	1.12	4.49	84.28	10.11	0.47
100	0.00	4.49	28.09	0.00	2.25	30.33	67.42	1.88
200	0.00	3.37	15.73	0.00	2.25	16.85	80.90	2.51
300	0.00	2.25	12.36	0.00	2.25	12.36	85.39	3.30
500	0.00	2.25	7.87	0.00	2.25	7.87	89.88	3.85
1000	0.00	2.25	3.37	0.00	2.25	3.37	94.38	7.43

ent iterations. S2M dominates clearly S1M in the very first iteration and after 200 iterations, while after the first few iterations S1M also manages to obtain good bounds. [Table 2](#) gives the comparison between these two algorithms at different times. For comparisons with $g = 1$, S1M is better than S2M only in at most 2.25% of the problems, while the converse varies between roughly 50% (at early times) and 8% (for late times). For $g = 5$, S2M still dominates S1M in most cases.

Sparse cuts yield better bounds than using solely the standard *PSD* cuts. The observed improvement is around 3% and 5% respectively for SPARSE1 and SPARSE2. When we are using the MINOR cuts, this value gets to 6% and 8% respectively for each type of sparsification algorithm used. [Table 3](#) compares PSDCUT (abbreviated by *S*) with S2M. The table shows that the sparse cuts generated by the sparsification procedures and minor *PSD* cuts yield better bounds than the standard cutting plane algorithm at fixed iterations. Comparisons performed at fixed times, on the other hand, show that considering the whole set of instances we do not get any improvement in the first 60 to 120 seconds of computation (see [Table 4](#)). Indeed S2M initially performs worse than the standard cutting plane algorithm, but after 60 to 120 seconds, it produces better bounds on average.

In Section 6 detailed computational results are given in [Tables 5](#) and 6 where for each instance we compare the duality gap closed by *S* and S2M at several iterations and times. The initial duality gap is obtained as in (4.1) as *RLT* – *OPT*. We then let S2M run with no time limit until the

TABLE 2
Comparison of S1M with S2M at several times.

Time	g = 1			g = 5			inc.	impr.
	S1M	S2M	Tie	S1M	S2M	Tie		
0.5	3.37	52.81	12.36	0.00	43.82	24.72	31.46	2.77
1	0.00	51.68	14.61	0.00	40.45	25.84	33.71	4.35
2	0.00	47.19	15.73	0.00	39.33	23.59	37.08	5.89
3	1.12	44.94	14.61	0.00	34.83	25.84	39.33	5.11
5	1.12	43.82	15.73	0.00	38.20	22.47	39.33	6.07
10	1.12	41.58	16.85	0.00	24.72	34.83	40.45	4.97
15	2.25	37.08	16.85	1.12	21.35	33.71	43.82	3.64
20	1.12	35.96	16.85	1.12	17.98	34.83	46.07	3.49
30	1.12	28.09	22.48	1.12	16.86	33.71	48.31	2.99
60	1.12	20.23	28.09	0.00	12.36	37.08	50.56	2.62
120	0.00	15.73	32.58	0.00	10.11	38.20	51.69	1.73
180	0.00	13.49	32.58	0.00	5.62	40.45	53.93	1.19
300	0.00	11.24	31.46	0.00	3.37	39.33	57.30	0.92
600	0.00	7.86	24.72	0.00	0.00	32.58	67.42	0.72

TABLE 3
Comparison of S with S2M at several iterations.

Iteration	g = 1			g = 5			inc.	impr.
	S	S2M	Tie	S	S2M	Tie		
1	0.00	76.40	23.60	0.00	61.80	38.20	0.00	10.47
2	0.00	84.27	15.73	0.00	55.06	44.94	0.00	10.26
3	0.00	83.15	16.85	0.00	48.31	51.69	0.00	10.38
5	0.00	80.90	19.10	0.00	40.45	59.55	0.00	10.09
10	1.12	71.91	26.97	0.00	41.57	58.43	0.00	8.87
15	1.12	60.67	38.21	1.12	35.96	62.92	0.00	7.49
20	1.12	53.93	40.45	1.12	29.21	65.17	4.50	6.22
30	1.12	34.83	53.93	0.00	16.85	73.03	10.12	5.04
50	1.12	25.84	62.92	0.00	13.48	76.40	10.12	3.75
100	1.12	8.99	21.35	0.00	5.62	25.84	68.54	5.57
200	0.00	5.62	8.99	0.00	3.37	11.24	85.39	7.66
300	0.00	3.37	7.87	0.00	3.37	7.87	88.76	8.86
500	0.00	3.37	5.62	0.00	3.37	5.62	91.01	8.72
1000	0.00	2.25	0.00	0.00	2.25	0.00	97.75	26.00

value s obtained does not improve by at least 0.01% over ten consecutive iterations. This value s is an upper bound on the value of the **PSD+RLT** relaxation. The column “bound” in the tables gives the value of $RLT - s$ as a percentage of the gap $RLT - OPT$, i.e. an approximation of the

TABLE 4
Comparison of S with S2M at several times.

Time	g = 1			g = 5			inc.	impr.
	S	S2M	Tie	S	S2M	Tie		
0.5	41.57	17.98	5.62	41.57	17.98	5.62	34.83	-9.42
1	41.57	14.61	5.62	39.33	13.48	8.99	38.20	-8.66
2	42.70	10.11	6.74	29.21	8.99	21.35	40.45	-8.73
3	41.57	8.99	8.99	31.46	6.74	21.35	40.45	-8.78
5	35.96	7.87	15.72	33.71	5.62	20.22	40.45	-7.87
10	34.84	7.87	13.48	30.34	4.50	21.35	43.81	-5.95
15	37.07	5.62	11.24	22.47	2.25	29.21	46.07	-5.48
20	37.07	5.62	8.99	17.98	1.12	32.58	48.32	-4.99
30	30.34	5.62	15.72	11.24	1.12	39.32	48.32	-3.9
60	11.24	12.36	25.84	11.24	2.25	35.95	50.56	-1.15
120	8.99	12.36	24.72	2.25	2.25	41.57	53.93	0.48
180	2.25	14.61	29.21	0.00	4.50	41.57	53.93	1.09
300	0.00	15.73	26.97	0.00	6.74	35.96	57.30	1.60
600	0.00	14.61	13.48	0.00	5.62	22.47	71.91	2.73

percentage of the gap closed by the **PSD+RLT** relaxation. The columns labeled S and S2M in the tables give the gap closed by the corresponding algorithms at different iterations.

Note that although S2M relies on numerous spectral decomposition computations, most of its running time is spent in generating cuts and reoptimization of the LP. For example, on the BoxQP instances with a time limit of 300 seconds, the average percentage of CPU time spent for obtaining spectral decompositions is below 21 for instances of size 30, below 15 for instances of size 40 and below 7 for instances of size 50.

5. Conclusions. This paper studies linearizations of the *PSD* cone based on spectral decompositions. Sparsification of eigenvectors corresponding to negative eigenvalues is shown to produce useful cuts in practice, in particular when the minor cuts are used. The goal of capturing most of the strength of a *PSD* relaxation through linear inequalities is achieved, although tailing off occurs relatively quickly. As an illustration of typical behavior of a *PSD* solver and our linear outer-approximation scheme, consider the two instances, spar020-100-1 and spar030-060-1, with respectively 20 and 30 variables. We use the SDP solver **SeDuMi** and S2M, keeping track at each iteration of the bound achieved and the time spent. [Figure 2](#) and [Figure 3](#) compare the bounds obtained by the two solvers at a given time. For the small size instance spar020-100-1, we note that S2M converges to the bound value more than twenty times faster than **SeDuMi**. In the medium size instance spar030-060-1 we note that S2M closes a large gap in the first ten to twenty iterations, and then tailing off occurs. To compute

the exact bound, SeDuMi requires 408 seconds while S2M requires 2,442 seconds to reach the same precision. Nevertheless, for our purposes, most of the benefits of the *PSD* constraints are captured in the early iterations.

Two additional improvements are possible. The first one is to use a cut separation procedure for the RLT inequalities, avoiding their inclusion in the initial LP and managing them as other cutting planes. This could potentially speed up the reoptimization of the LP. Another possibility is to use a mix of the S and S2M algorithms, using the former in the early iterations and then switching to the latter.

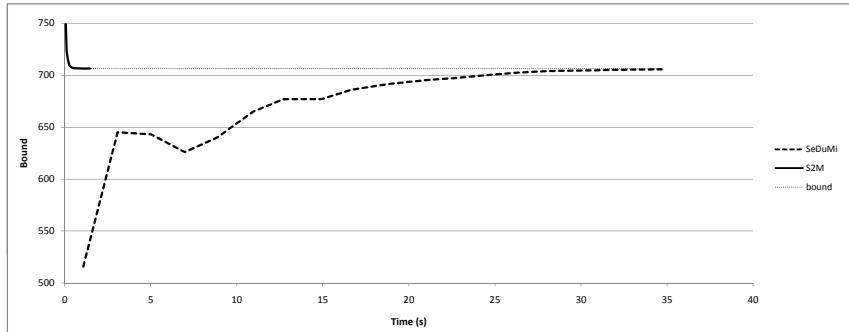


FIG. 2. *Instance spar020-100-1.*

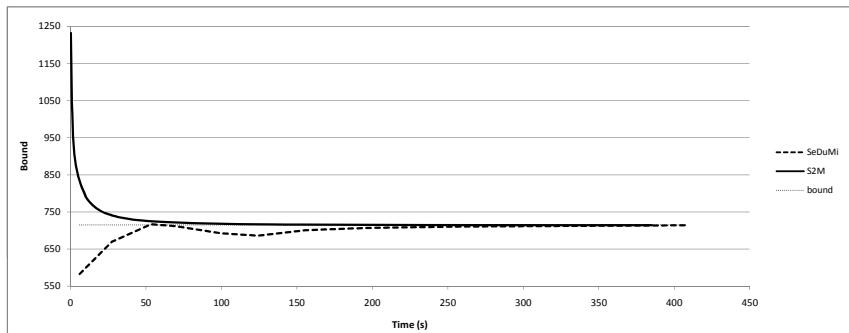


FIG. 3. *Instance spar030-060-1.*

Acknowledgments. The authors warmly thank Anureet Saxena for the useful discussions that led to the results obtained in this work.

6. Appendix.

TABLE 5
Duality gap closed at several iterations for each instance.

Instance	$ x $	$ y $	bound	iter. 2			iter. 10			iter. 50		
				S	S2M	S	S	S2M	S	S	S2M	S
circle	3	0	45.79	0.00	0.00	10.97	41.31	45.77	45.79			
dispatch	3	1	100.00	25.59	27.92	37.25	35.76	95.90	92.17			
ex2.1.10	20	0	22.05	3.93	8.65	15.93	21.05	22.05	22.05			
ex3.1.2	5	0	49.75	49.75	49.75	49.75	49.75	49.75	49.75			
ex4.1.1	3	0	100.00	99.81	99.84	100.00	100.00	100.00	100.00			
ex4.1.3	3	0	56.40	0.00	0.00	51.19	51.19	56.40	56.40			
ex4.1.4	3	0	100.00	22.33	42.78	98.98	99.98	100.00	100.00			
ex4.1.6	3	0	100.00	69.44	69.87	92.62	99.94	100.00	100.00			
ex4.1.7	3	0	100.00	18.00	48.17	96.86	99.90	100.00	100.00			
ex4.1.8	3	0	100.00	56.90	81.93	99.76	99.93	100.00	100.00			
ex8.1.4	4	0	100.00	94.91	95.19	99.98	100.00	100.00	100.00			
ex8.1.5	5	0	68.26	32.32	39.17	59.01	66.76	68.00	68.25			
ex8.1.7	9	0	77.43	3.04	33.75	33.13	53.44	64.03	75.38			
ex8.4.1	21	1	91.81	4.45	21.80	18.60	45.08	38.07	69.83			
ex9.2.1	10	0	54.52	0.00	42.55	0.01	50.13	0.01	51.90			
ex9.2.2	10	0	70.37	0.00	14.08	2.34	51.97	7.12	69.41			
ex9.2.4	6	2	99.87	0.00	24.84	25.24	99.85	86.37	99.87			
ex9.2.6	16	0	99.88	3.50	99.42	23.09	99.86	62.32	99.88			
ex9.2.7	10	0	42.30	0.00	4.59	0.00	27.34	3.14	34.91			
himmel11	5	4	49.75	49.75	49.75	49.75	49.75	49.75	49.75			
hydro	12	19	52.06	0.00	20.87	21.95	29.03	26.04	31.39			
mathopt1	4	0	100.00	95.76	100.00	99.96	100.00	100.00	100.00			
mathopt2	3	0	100.00	99.84	99.93	100.00	100.00	100.00	100.00			
meanvar	7	1	100.00	0.00	0.00	78.35	95.84	100.00	100.00			
nemhaus	5	0	53.97	26.00	26.41	48.49	50.16	53.87	53.96			
prob06	2	0	100.00	90.61	92.39	98.39	98.39	98.39	98.39			
prob09	3	1	100.00	0.00	99.00	61.14	99.96	99.64	100.00			
process	9	3	8.00	0.00	4.25	0.00	4.98	5.73	5.73			
qp1	50	0	100.00	79.59	89.09	93.89	99.77	98.93	100.00			
qp2	50	0	100.00	55.94	70.99	82.42	93.92	93.04	99.35			
rblock	3	0	100.00	97.48	100.00	99.96	100.00	100.00	100.00			

Continued on Next Page...

Table 5 (Continued)

Instance	$ x $	$ y $	bound	iter. 2		iter. 10		iter. 50
				S	S2M	S	S2M	
st_e10	3	1	100.00	56.90	81.93	99.76	99.93	100.00
st_e18	2	0	100.00	0.00	0.00	98.72	98.72	100.00
st_e19	3	1	93.51	5.14	15.93	29.97	60.10	93.50
st_e25	4	0	87.55	55.80	55.80	87.02	87.01	87.23
st_e28	5	4	49.75	49.75	49.75	49.75	49.75	49.75
st_lqpblk1	8	0	97.99	71.99	76.69	97.20	97.95	97.99
st_lqpblk2	8	0	97.93	70.55	75.16	94.93	97.52	97.93
spar020-100-1	20	0	100.00	91.15	94.64	99.77	99.99	100.00
spar020-100-2	20	0	99.70	90.12	92.64	98.17	99.32	99.66
spar020-100-3	20	0	100.00	96.96	98.51	100.00	100.00	99.69
spar030-060-1	30	0	98.87	43.53	53.64	79.61	87.39	93.90
spar030-060-2	30	0	100.00	80.74	89.73	99.89	100.00	100.00
spar030-060-3	30	0	99.40	67.43	71.94	91.48	95.68	98.75
spar030-070-1	30	0	97.99	49.05	54.94	76.54	86.51	91.15
spar030-070-2	30	0	100.00	81.19	85.82	99.26	99.99	100.00
spar030-070-3	30	0	99.98	85.97	87.43	98.44	99.52	99.97
spar030-080-1	30	0	98.99	64.44	70.99	87.32	92.11	96.23
spar030-080-2	30	0	100.00	92.78	95.45	100.00	100.00	100.00
spar030-080-3	30	0	100.00	92.71	94.18	99.99	100.00	100.00
spar030-090-1	30	0	100.00	80.37	86.35	97.27	99.30	100.00
spar030-090-2	30	0	100.00	86.09	89.26	98.13	99.65	100.00
spar030-090-3	30	0	100.00	90.65	91.56	99.97	100.00	100.00
spar030-100-1	30	0	100.00	77.28	83.25	95.20	98.30	99.85
spar030-100-2	30	0	99.96	76.78	81.65	93.44	96.84	98.70
spar030-100-3	30	0	99.85	86.82	88.74	97.45	98.75	99.75
spar040-030-1	40	0	100.00	25.60	41.96	73.59	84.72	99.13
spar040-030-2	40	0	100.00	30.93	53.39	79.34	95.62	99.46
spar040-030-3	40	0	100.00	9.21	31.38	66.46	86.62	98.53
spar040-040-1	40	0	96.74	23.62	29.03	63.04	75.93	85.93
spar040-040-2	40	0	100.00	33.17	48.87	89.08	97.94	100.00
spar040-040-3	40	0	99.18	21.77	30.31	70.44	80.96	91.37
spar040-050-1	40	0	99.42	35.62	44.87	73.11	84.05	92.81
spar040-050-2	40	0	100.00	41.91	51.72	82.38	91.27	97.26
spar040-050-3	40	0	100.00	46.22	52.89	84.04	90.70	96.88
spar040-060-1	40	0	98.09	46.22	52.89	81.65	87.28	95.97

Continued on Next Page...

Table 5 (Continued)

Instance	$ x $	$ y $	bound	iter. 2		iter. 10		iter. 50	
				S	S2M	S	S2M	S	S2M
spar040-060-2	40	0	100.00	63.02	72.87	94.09	97.66	99.78	100.00
spar040-060-3	40	0	100.00	78.09	87.91	99.30	99.99	100.00	100.00
spar040-070-1	40	0	100.00	64.02	71.33	93.92	97.35	99.77	100.00
spar040-070-2	40	0	100.00	67.49	76.78	95.12	97.97	99.97	100.00
spar040-070-3	40	0	100.00	70.13	79.43	95.65	97.99	99.75	100.00
spar040-080-1	40	0	100.00	63.06	69.40	91.09	95.44	99.00	99.97
spar040-080-2	40	0	100.00	71.42	79.77	94.98	97.62	99.92	100.00
spar040-080-3	40	0	99.99	83.93	88.65	97.76	98.86	99.81	99.95
spar040-090-1	40	0	100.00	75.73	79.96	95.34	97.43	99.46	99.91
spar040-090-2	40	0	99.97	76.39	80.97	95.16	96.72	99.20	99.81
spar040-090-3	40	0	100.00	84.90	87.04	98.33	99.52	100.00	100.00
spar040-100-1	40	0	100.00	87.64	90.43	98.27	99.35	99.98	100.00
spar040-100-2	40	0	99.87	79.78	83.02	94.58	96.76	98.74	99.50
spar040-100-3	40	0	98.70	72.69	78.31	90.83	93.03	95.84	97.36
spar050-030-1	50	0	100.00	3.11	17.60	58.23	79.98	-	-
spar050-030-2	50	0	99.27	1.35	16.67	51.11	70.58	-	-
spar050-030-3	50	0	99.29	0.08	13.63	50.19	67.46	-	-
spar050-040-1	50	0	100.00	23.13	30.86	72.10	81.73	-	-
spar050-040-2	50	0	99.39	21.89	34.45	71.24	81.63	-	-
spar050-040-3	50	0	100.00	27.18	37.42	83.96	91.70	-	-
spar050-050-1	50	0	93.02	25.24	33.77	61.42	68.75	-	-
spar050-050-2	50	0	98.74	32.10	41.26	77.48	83.48	-	-
spar050-050-3	50	0	98.84	38.57	44.67	80.97	85.36	-	-
Average	-	-	-	48.75	59.00	75.53	84.39	85.85	89.60

TABLE 6
Duality gap closed at several times for each instance. (Instances solved in less than 1 second are not shown.)

Instance	bound	1 s		60 s		180 s		300 s		600 s	
		S	S2M	S	S2M	S	S2M	S	S2M	S	S2M
ex4_1_4	100.00	-	100.00	-	100.00	-	100.00	-	100.00	-	-
ex8_1_4	100.00	-	100.00	-	100.00	-	100.00	-	100.00	-	-
ex8_1_7	77.43	77.43	77.37	77.37	77.37	-	-	-	-	-	-
ex8_4_1	91.81	28.14	36.24	61.60	90.43	-	-	-	-	-	-
ex9_2_2	70.37	-	70.35	-	-	-	-	-	-	-	-
ex9_2_6	98.88	96.28	-	-	-	-	-	-	-	-	-
hydro	52.06	26.43	31.46	-	-	-	-	-	-	-	-
mathopt2	100.00	-	100.00	-	-	-	-	-	-	-	-
process	8.00	-	7.66	-	-	-	-	-	-	-	-
qp1	100.00	79.99	80.28	98.22	99.52	99.73	99.96	99.92	99.98	99.99	100.00
qp2	100.00	55.82	55.27	91.74	95.56	95.86	98.69	97.41	99.66	98.80	100.00
spar020-100-1	100.00	100.00	100.00	-	-	-	-	-	-	-	-
spar020-100-2	99.70	99.67	99.61	-	-	-	-	-	-	-	-
spar020-100-3	100.00	-	100.00	-	-	-	-	-	-	-	-
spar030-060-1	98.87	69.98	58.72	96.53	97.61	98.45	98.70	98.68	98.82	-	-
spar030-060-2	100.00	96.52	91.05	-	-	-	-	-	-	-	-
spar030-060-3	99.40	82.99	76.15	99.27	99.32	99.38	99.39	99.39	99.40	99.40	99.40
spar030-070-1	97.99	69.81	60.36	94.50	96.38	97.29	97.73	97.70	97.91	-	97.98
spar030-070-2	100.00	96.05	87.93	-	-	-	-	-	-	-	-
spar030-070-3	99.98	96.26	90.42	99.98	99.98	99.98	99.98	99.98	99.98	-	-
spar030-080-1	98.99	83.36	74.42	97.80	98.11	98.74	98.88	98.89	98.96	-	98.99
spar030-080-2	100.00	99.83	96.70	-	-	-	-	-	-	-	-
spar030-080-3	100.00	99.88	95.87	-	-	-	-	-	-	-	-
spar030-090-1	100.00	92.86	87.69	-	-	-	-	-	-	-	-
spar030-090-2	100.00	93.80	88.46	-	100.00	-	-	-	-	-	-
spar030-090-3	100.00	97.78	91.35	-	-	-	-	-	-	-	-
spar030-100-1	100.00	91.04	84.34	100.00	-	-	-	-	-	-	-
spar030-100-2	99.96	90.21	83.14	99.56	99.75	99.91	99.95	99.95	99.96	-	-
spar030-100-3	98.85	94.26	89.55	99.84	99.84	99.85	99.85	99.85	99.85	99.85	99.85
spar040-030-1	100.00	28.97	40.51	89.30	84.19	99.06	99.98	99.98	100.00	-	100.00

Continued on Next Page...

Table 6 (Continued)

Instance	bound	1 s		60 s		180 s		300 s		600 s	
		S	S2M	S	S2M	S	S2M	S	S2M	S	S2M
spar040-030-2	100.00	31.97	48.01	94.01	96.39	99.58	99.98	99.99	100.00	100.00	-
spar040-030-3	100.00	9.20	27.59	81.66	85.43	97.25	99.86	99.81	100.00	100.00	-
spar040-040-1	96.74	19.38	22.90	70.35	75.45	80.73	88.63	85.34	92.29	90.79	94.74
spar040-040-2	100.00	24.51	29.87	98.63	98.60	100.00	100.00	-	-	-	-
spar040-040-3	99.18	20.88	21.31	78.28	79.31	86.02	91.22	89.52	95.04	94.07	97.71
spar040-050-1	99.42	28.96	21.27	80.18	84.01	88.70	94.62	92.75	96.71	96.53	98.32
spar040-050-2	95.48	29.52	16.91	91.33	91.42	97.01	97.97	98.26	98.87	-	99.31
spar040-050-3	100.00	28.67	19.81	90.03	90.72	95.68	97.51	97.49	99.08	98.92	99.89
spar040-060-1	98.09	37.16	17.10	86.26	87.13	90.18	93.50	92.25	95.32	95.05	96.84
spar040-060-2	100.00	39.57	22.83	98.09	98.22	99.90	99.96	100.00	100.00	100.00	-
spar040-060-3	100.00	52.41	30.57	100.00	99.99	-	-	-	-	-	-
spar040-070-1	100.00	50.01	21.79	97.74	97.78	99.80	99.87	99.97	99.99	100.00	100.00
spar040-070-2	100.00	47.57	25.19	98.81	98.46	99.99	99.99	100.00	-	-	-
spar040-070-3	100.00	47.22	21.95	98.96	98.70	99.88	99.92	99.98	100.00	100.00	100.00
spar040-080-1	100.00	51.66	28.00	95.13	95.38	98.29	99.05	99.09	99.74	99.77	99.99
spar040-080-2	100.00	52.24	25.94	98.71	98.31	99.95	99.97	100.00	100.00	-	-
spar040-080-3	99.99	56.05	26.98	99.54	99.25	99.89	99.88	99.94	99.95	99.97	99.98
spar040-090-1	100.00	59.71	28.17	98.10	97.86	99.43	99.61	99.70	99.86	99.90	99.99
spar040-090-2	99.97	59.14	29.82	97.83	97.70	99.34	99.58	99.68	99.81	99.86	99.93
spar040-090-3	100.00	63.07	34.62	99.94	99.85	100.00	100.00	-	-	-	-
spar040-100-1	100.00	69.47	28.24	99.66	99.47	99.99	99.99	100.00	100.00	-	-
spar040-100-2	99.87	65.27	26.07	97.34	96.87	98.60	98.98	99.02	99.39	99.44	99.69
spar040-100-3	98.70	61.40	29.61	93.01	93.17	94.91	96.02	95.81	97.00	96.84	97.77
spar050-030-1	100.00	0.37	3.63	54.46	37.52	70.10	73.34	76.87	84.75	86.23	96.33
spar050-030-2	99.27	0.08	2.79	44.68	38.62	59.58	64.94	67.79	74.98	77.02	86.58
spar050-030-3	99.29	0.0	2.75	44.32	32.31	57.13	59.07	62.54	68.99	71.18	82.86
spar050-040-1	100.00	3.76	1.77	69.97	56.87	77.15	78.30	80.31	84.30	84.90	91.79
spar050-040-2	99.39	2.08	2.84	68.64	58.47	77.72	77.61	81.54	83.63	86.40	90.94
spar050-040-3	100.00	1.76	2.31	79.44	65.71	89.73	87.74	92.67	93.00	95.99	97.69
spar050-050-1	93.02	4.91	1.84	60.64	53.28	65.52	66.42	66.81	70.38	68.45	74.76
spar050-050-2	98.74	6.18	3.39	76.56	68.33	82.34	82.21	84.94	86.52	-	91.34
spar050-050-3	98.84	6.12	2.82	79.38	69.23	84.95	83.23	86.98	86.98	89.77	91.57
Average	-	51.45	42.96	87.50	86.38	92.14	93.22	93.18	94.77	93.16	95.86

REFERENCES

- [1] F.A. AL-KHAYYAL AND J.E. FALK, Jointly constrained biconvex programming. *Math. Oper. Res.* **8**, pp. 273–286, 1983.
- [2] K.M. ANSTREICHER, Semidefinite Programming versus the Reformulation-Linearization Technique for Nonconvex Quadratically Constrained Quadratic Programming, *Journal of Global Optimization*, **43**, pp. 471–484, 2009.
- [3] E. BALAS, Disjunctive programming: properties of the convex hull of feasible points. *Disc. Appl. Math.* **89**, 1998.
- [4] M.S. BAZARAA, H.D. SHERALI, AND C.M. SHETTY, Nonlinear Programming: Theory and Algorithms. Wiley, 2006.
- [5] S. BOYD AND L. VANDENBERGHE, Convex Optimization. Cambridge University Press, 2004.
- [6] S.A. BURER AND A.N. LETCHFORD, On Non-Convex Quadratic Programming with Box Constraints, *SIAM Journal on Optimization*, **20**, pp. 1073–1089, 2009.
- [7] B. BORCHERS, CSDP, A C Library for Semidefinite Programming, *Optimization Methods and Software* **11**(1), pp. 613–623, 1999.
- [8] COmputational INfrastructure for Operations Research (COIN-OR). <http://www.coin-or.org>.
- [9] S.J. BENSON AND Y. YE, DSDP5: Software For Semidefinite Programming. Available at <http://www-unix.mcs.anl.gov/DSDP>.
- [10] GamsWorld Global Optimization library, <http://www.gamsworld.org/global/globallib/globalstat.htm>.
- [11] L. LOVÁSZ AND A. SCHRIJVER, Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, May 1991.
- [12] G.P. MCCORMICK, Nonlinear programming: theory, algorithms and applications. John Wiley & sons, 1983.
- [13] <http://www.andrew.cmu.edu/user/aqualizz/research/MIQCP>.
- [14] A. SAXENA, P. BONAMI, AND J. LEE, Convex Relaxations of Non-Convex Mixed Integer Quadratically Constrained Programs: Extended Formulations, *Mathematical Programming*, Series B, **124**(1–2), pp. 383–411, 2010.
- [15] ———, Convex Relaxations of Non-Convex Mixed Integer Quadratically Constrained Programs: Projected Formulations, Optimization Online, November 2008. Available at http://www.optimization-online.org/DB_HTML/2008/11/2145.html.
- [16] J.F. STURM, SeDuMi: An Optimization Package over Symmetric Cones. Available at <http://sedumi.mcmaster.ca>.
- [17] H.D. SHERALI AND W.P. ADAMS, A reformulation-linearization technique for solving discrete and continuous nonconvex problems. Kluwer, Dordrecht 1998.
- [18] H.D. SHERALI AND B.M.P. FRATICELLI, Enhancing RLT relaxations via a new class of semidefinite cuts. *J. Global Optim.* **22**, pp. 233–261, 2002.
- [19] N.Z. SHOR, Quadratic optimization problems. *Tekhnicheskaya Kibernetika*, 1, 1987.
- [20] K. SIVARAMAKRISHNAN AND J. MITCHELL, Semi-infinite linear programming approaches to semidefinite programming (SDP) problems. Novel Approaches to Hard Discrete Optimization, edited by P.M. Pardalos and H. Wolkowicz, *Fields Institute Communications Series, American Math. Society*, 2003.
- [21] ———, Properties of a cutting plane method for semidefinite programming, Technical Report, Department of Mathematics, North Carolina State University, September 2007.
- [22] K.C. TOH, M.J. TODD, AND R.H.TÜTÜNCÜ, SDPT3: A MATLAB software for semidefinite-quadratic-linear programming. Available at <http://www.math.nus.edu.sg/~mattohkc/sdpt3.html>.

- [23] L. VANDENBERGHE AND S. BOYD, Semidefinite Programming. *SIAM Review* **38**(1), pp. 49–95, 1996.
- [24] D. VANDENBUSSCHE AND G. L. NEMHAUSER, A branch-and-cut algorithm for non-convex quadratic programs with box constraints. *Math. Prog.* **102**(3), pp. 559–575, 2005.
- [25] H. WOLKOWICZ, R. SAIGAL, AND L. VANDENBERGHE, *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*. Springer, 2000.

EXTENDING A CIP FRAMEWORK TO SOLVE MIQCPs*

TIMO BERTHOLD[†], STEFAN HEINZ[†], AND STEFAN VIGERSKE[‡]

Abstract. This paper discusses how to build a solver for mixed integer quadratically constrained programs (MIQCPs) by extending a framework for constraint integer programming (CIP). The advantage of this approach is that we can utilize the full power of advanced MILP and CP technologies, in particular for the linear relaxation and the discrete components of the problem. We use an outer approximation generated by linearization of convex constraints and linear underestimation of nonconvex constraints to relax the problem. Further, we give an overview of the reformulation, separation, and propagation techniques that are used to handle the quadratic constraints efficiently.

We implemented these methods in the branch-cut-and-price framework **SCIP**. Computational experiments indicating the potential of the approach and evaluating the impact of the algorithmic components are provided.

Key words. Mixed integer quadratically constrained programming, constraint integer programming, branch-and-cut, convex relaxation, domain propagation, primal heuristic, nonconvex.

AMS(MOS) subject classifications. 90C11, 90C20, 90C26, 90C27, 90C57.

1. Introduction. In recent years, substantial progress has been made in the solvability of generic *mixed integer linear programs (MILPs)* [2, 12]. Furthermore, it has been shown that successful MILP solving techniques can often be extended to the more general case of *mixed integer nonlinear programs (MINLPs)* [1, 6, 13]. Analogously, several authors have shown that an integrated approach of *constraint programming (CP)* and MILP can help to solve optimization problems that were intractable with either of the two methods alone, for an overview see [17].

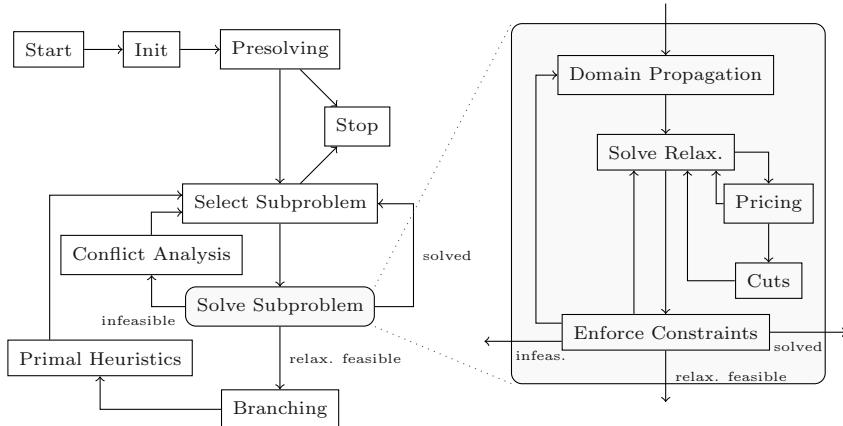
The paradigm of *constraint integer programming (CIP)* [2, 4] combines modeling and solving techniques from the fields of constraint programming (CP), mixed integer programming, and *satisfiability testing (SAT)*. The concept of CIP aims at restricting the generality of CP modeling as little as needed while still retaining the full performance of MILP solving techniques. Such a paradigm allows us to address a wide range of optimization problems. For example, in [2], it is shown that CIP includes MILP and constraint programming over finite domains as special cases.

The goal of this paper is to show, how a framework for CIPs can be extended towards a competitive solver for *mixed integer quadratically constrained programs (MIQCPs)*, which are an important subclass of MINLPs. This framework allows us to utilize the power of already existing MILP and

*Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin, <http://www.matheon.de>.

[†]Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany ({berthold, heinz}@zib.de).

[‡]Humboldt University Berlin, Unter den Linden 6, 10099 Berlin, Germany (stefan@math.hu-berlin.de).

FIG. 1. *Flowchart of the main solving loop of SCIP.*

CP technologies for handling the linear and the discrete parts of the problem. The integration of MIQCP is a first step towards the incorporation of MINLP into the concept of constraint integer programming.

We extended the branch-cut-and-price framework **SCIP** (Solving Constraint Integer Programs) [2, 3] by adding methods for MIQCP. **SCIP** incorporates the idea of CIP and implements several state-of-the-art techniques for solving MILPs. Due to its plugin-based design, it can be easily customized, e.g., by adding problem specific separation, presolving, or domain propagation algorithms.

The framework **SCIP** solves CIPs by a branch-and-bound algorithm. The problem is recursively split into smaller subproblems, thereby creating a search tree and implicitly enumerating all potential solutions. At each subproblem, domain propagation is performed to exclude further values from the variables' domains, and a relaxation may be solved to achieve a local lower bound – assuming the problem is to minimize the objective function. The relaxation may be strengthened by adding further valid constraints (e.g., linear inequalities), which cut off the optimal solution of the relaxation. In case a subproblem is found to be infeasible, conflict analysis is performed to learn additional valid constraints. Primal heuristics are used as supplementary methods to improve the upper bound. Figure 1 illustrates the main algorithmic components of **SCIP**. In the context of this article, the relaxation employed in **SCIP** is a linear program (LP).

The remainder of this article is organized as follows. In Section 2, we formally define MIQCP and CIP, in Sections 3, 4, and 5, we show how to handle quadratic constraints inside **SCIP**, and in Section 6, we present computational results.

2. Problem definition. An MIQCP is an optimization problem of the form

$$\begin{aligned} \min \quad & d^T x \\ \text{s.t.} \quad & x^T A_i x + b_i^T x + c_i \leq 0 && \text{for } i = 1, \dots, m \\ & x_k^L \leq x_k \leq x_k^U && \text{for all } k \in N \\ & x_k \in \mathbb{Z} && \text{for all } k \in I, \end{aligned} \tag{2.1}$$

where $I \subseteq N := \{1, \dots, n\}$ is the index set of the integer variables, $d \in \mathbb{Q}^n$, $A_i \in \mathbb{Q}^{n \times n}$ and symmetric, $b_i \in \mathbb{Q}^n$, $c_i \in \mathbb{Q}$ for $i = 1, \dots, m$, $x^L \in \overline{\mathbb{Q}}^n$ and $x^U \in \overline{\mathbb{Q}}^n$, with $\overline{\mathbb{Q}} := \mathbb{Q} \cup \{\pm\infty\}$, are the lower and upper bounds of the variables x , respectively (\mathbb{Q} denotes the rational numbers). Note that we do not require the matrices A_i to be positive semidefinite, hence we also allow for nonconvex quadratic constraints. If $I = \emptyset$, we call (2.1) a *quadratically constrained program (QCP)*.

The definition of CIP, as given in [2, 4], requires a linear objective function. This is, however, just a technical prerequisite, as a quadratic (or more general) objective $f(x)$ can be modeled by introducing an auxiliary objective variable z that is linked to the actual nonlinear objective function with a constraint $f(x) \leq z$. Thus, formulation (2.1) also covers the general case of mixed integer quadratically constrained quadratic problems.

In this article, we use a definition of CIP which is slightly different from the one given in [2, 4]. A constraint integer program consists of solving

$$\begin{aligned} \min \quad & d^T x \\ \text{s.t.} \quad & \mathcal{C}_i(x) = 1 && \text{for } i = 1, \dots, m \\ & x_k \in \mathbb{Z} && \text{for all } k \in I, \end{aligned}$$

with a finite set of constraints $\mathcal{C}_i : \mathbb{Q}^n \rightarrow \{0, 1\}$, for $i = 1, \dots, m$, the index set $I \subseteq N$ of the integer variables, and an objective function vector $d \in \mathbb{Q}^n$.

In [2, 4], it is required that the subproblem remaining after fixing all integer variables be a linear program in order to guarantee termination in finite time. In this article, we, however, require a subproblem with all integer variables fixed to be a QCP. Note that, using *spatial* branch-and-bound algorithms, QCPs with finite bounds on the variables can be solved in finite time up to a given tolerance [18].

3. A constraint handler for quadratic constraints. In SCIP, a constraint handler defines the semantics and the algorithms to process constraints of a certain class. A single constraint handler is responsible for all the constraints belonging to its constraint class. Each constraint handler has to implement an enforcement method. In enforcement, the handler has to decide whether a given solution, e.g., the optimum of a relaxation¹, sat-

¹For this section, we assume that the LP relaxation is bounded. In our implementation, the so-called pseudo solution, see [2, 3] for details, will be used in the case of unbounded LP relaxations.

isfies all of its constraints. If the solution violates one or more constraints, the handler may resolve the infeasibility by adding another constraint, performing a domain reduction, or a branching.

For speeding up computation, a constraint handler may further implement additional features like presolving, cutting plane separation, and domain propagation for its particular class of constraints. Besides that, a constraint handler can add valid linear inequalities to the initial LP relaxation. For example, all constraint handler for (general or specialized) linear constraints add their constraints to the initial LP relaxation. The constraint handler for quadratic constraints adds one linear inequality that is obtained by the method given in Section 3.2 below.

In the following, we discuss the presolving, separation, propagation, and enforcement algorithms that are used to handle quadratic constraints.

3.1. Presolving. During the presolving phase, a set of reformulations and simplifications are tried. If **SCIP** fixes or aggregates variables, e.g., using global presolving methods like dual bound reduction [2], then the corresponding reformulations will also be realized in the quadratic constraints. Bounds on the variables are tightened using the domain propagation method described in Section 3.3. If, due to reformulations, the quadratic part of a constraint vanishes, it is replaced by the corresponding linear constraint. Furthermore, the following reformulations are performed.

Binary Variables. A square of a binary variable is replaced by the binary variable itself. Further, if a constraint contains a product of a binary variable with a linear term, i.e., $x \sum_{i=1}^k a_i y_i$, where x is a binary variable, y_i are variables with finite bounds, and $a_i \in \mathbb{Q}$, $i = 1, \dots, k$, then this product will be replaced by a new variable $z \in \mathbb{R}$ and the linear constraints

$$\begin{aligned} y^L x &\leq z \leq y^U x \\ \sum_{i=1}^k a_i y_i - y^U(1-x) &\leq z \leq \sum_{i=1}^k a_i y_i - y^L(1-x), \text{ where} \\ y^L &:= \sum_{\substack{i=1, \\ a_i > 0}}^k a_i y_i^L + \sum_{\substack{i=1, \\ a_i < 0}}^k a_i y_i^U, \text{ and} \\ y^U &:= \sum_{\substack{i=1, \\ a_i > 0}}^k a_i y_i^U + \sum_{\substack{i=1, \\ a_i < 0}}^k a_i y_i^L. \end{aligned} \quad (3.1)$$

In the case that $k = 1$ and y_1 is also a binary variable, the product xy_1 can also be handled by **SCIP**'s handler for AND constraints [11].

Second-Order Cone (SOC) constraints. Constraints of the form

$$\gamma + \sum_{i=1}^k (\alpha_i(x_i + \beta_i))^2 \leq (\alpha_0(y + \beta_0))^2, \quad (3.2)$$

with $k \geq 2$, $\alpha_i, \beta_i \in \mathbb{Q}$, $i = 0, \dots, k$, $\gamma \in \mathbb{Q}_+$, and $y^L \geq -\beta_0$ are recognized as SOC constraints and handled by a specialized constraint handler, cf. Section 4.

Convexity. After the presolving phase, each quadratic function is checked for convexity by computing the sign of the minimum eigenvalue of the coefficient matrix A . This information will be used for separation.

3.2. Separation. If the current LP solution \tilde{x} violates some constraints, a constraint handler may add valid cutting planes in order to strengthen the formulation.

For a violated convex constraint i , this is always possible by linearizing the constraint function at \tilde{x} . Thus, we add the valid inequality

$$c_i - \tilde{x}^T A_i \tilde{x} + (b_i^T + 2\tilde{x}^T A_i)x \leq 0 \quad (3.3)$$

to separate \tilde{x} . In the important special case that $x^T A_i x \equiv ax_j^2$ for some $a > 0$ and $j \in I$ with $\tilde{x}_j \notin \mathbb{Z}$, we generate the cut

$$c_i + b_i^T x + a(2\lfloor \tilde{x}_j \rfloor + 1)x_j - a\lfloor \tilde{x}_j \rfloor \lceil \tilde{x}_j \rceil \leq 0, \quad (3.4)$$

which is obtained by underestimating $x_j \in \mathbb{Z} \mapsto x_j^2$ by the secant defined by the points $(\lfloor \tilde{x}_j \rfloor, \lfloor \tilde{x}_j \rfloor^2)$ and $(\lceil \tilde{x}_j \rceil, \lceil \tilde{x}_j \rceil^2)$. Note that the violation of (3.4) by \tilde{x} is larger than that of (3.3).

For a violated nonconvex constraint i , we currently underestimate each term of $x^T A_i x$ separately. A term ax_j^2 with $a > 0$, $j \in N$, is underestimated as just discussed. For the case $a < 0$, however, the tightest linear underestimation for the term ax_j^2 is given by the secant approximation $a(x_j^L + x_j^U)x_j - ax_j^L x_j^U$, if x_j^L and x_j^U are finite. Otherwise, if $x_j^L = -\infty$ or $x_j^U = \infty$, we skip separation for constraint i . For a bilinear term $ax_j x_k$ with $a > 0$, we utilize the McCormick underestimators [21]

$$\begin{aligned} ax_j x_k &\geq ax_j^L x_k + ax_k^L x_j - ax_j^L x_k^L, \\ ax_j x_k &\geq ax_j^U x_k + ax_k^U x_j - ax_j^U x_k^U. \end{aligned}$$

If $(x_j^U - x_j^L)\tilde{x}_k + (x_k^U - x_k^L)\tilde{x}_j \leq x_j^U x_k^U - x_j^L x_k^L$ and the bounds x_j^L and x_k^L are finite, the former is used for cut generation, elsewise the latter is used. If both x_j^L or x_k^L and x_j^U or x_k^U are infinite, we skip separation for constraint i . Similar, for a bilinear term $ax_j x_k$ with $a < 0$, the McCormick underestimators are

$$\begin{aligned} ax_j x_k &\geq ax_j^U x_k + ax_k^L x_j - ax_j^U x_k^L, \\ ax_j x_k &\geq ax_j^L x_k + ax_k^U x_j - ax_j^L x_k^U. \end{aligned}$$

If $(x_j^U - x_j^L)\tilde{x}_k - (x_k^U - x_k^L)\tilde{x}_j \leq x_j^U x_k^L - x_j^L x_k^U$ and the bounds x_j^U and x_k^L are finite, the former is used for cut generation, elsewise the latter is used.

In the case that a linear inequality generated by this method does not cut off the current LP solution \tilde{x} , the infeasibility has to be resolved in enforcement, see Section 3.4. Besides others, the enforcement method may apply a spatial branching operation on a variable x_j , creating two subproblems, which both contain a strictly smaller domain for x_j . This results in tighter linear underestimators.

3.3. Propagation. In the domain propagation call, a constraint handler may deduce new restrictions upon local domains of variables. Such deductions may yield stronger linear underestimators in the separation procedures, prune nodes due to infeasibility of a constraint, or result in further deductions for other constraints. For quadratic constraints, we implemented an interval-arithmetic based method similar to [16]. To allow for an efficient propagation, we write a quadratic constraint in the form

$$\sum_{j \in J} d_j x_j + \sum_{k \in K} \left(e_k + p_{k,k} x_k + \sum_{r \in K} p_{k,r} x_r \right) x_k \in [\ell, u], \quad (3.5)$$

such that $d_j, e_k, p_{k,r} \in \mathbb{Q}$, $\ell, u \in \overline{\mathbb{Q}}$, $J \cup K \subseteq N$, $J \cap K = \emptyset$, and $p_{k,r} = 0$ for $k > r$. For a given $a \in \mathbb{Q}$, an interval $[b^L, b^U]$, and a variable y with domain $[y^L, y^U]$, we denote by $q(a, [b^L, b^U], y)$ the set $\{by + ay^2 : y \in [y^L, y^U], b \in [b^L, b^U]\}$. This set can be computed analytically [16].

The forward propagation step aims at tightening the bounds $[\ell, u]$ in (3.5). For this purpose, we replace the variables x_j and x_r in (3.5) by their domain to obtain the “interval-equation”

$$\sum_{j \in J} d_j [x_j^L, x_j^U] + \sum_{k \in K} ([f_k^L, f_k^U] x_k + p_{k,k} x_k^2) \in [\ell, u],$$

where $[f_k^L, f_k^U] := [e_k, e_k] + \sum_{r \in K} p_{k,r} [x_r^L, x_r^U]$. Computing $[h^L, h^U] := \sum_{j \in J} d_j [x_j^L, x_j^U] + \sum_{k \in K} q(p_{k,k}, [f_k^L, f_k^U], x_k)$ yields an interval that contains all values that the left hand side of (3.5) can take w.r.t. the current variables’ domains. If $[h^L, h^U] \cap [\ell, u] = \emptyset$, then (3.5) cannot be satisfied for any $x \in [x^L, x^U]$ and the current branch-and-bound node can be pruned. Otherwise, the interval $[\ell, u]$ can be tightened to $[\ell, u] \cap [h^L, h^U]$.

The backward propagation step aims at inferring domain deductions on the variables in (3.5) using the interval $[\ell, u]$. For a “linear” variable x_j , $j \in J$, we can easily infer the bounds

$$\frac{1}{d_j} \left([\ell, u] - \sum_{j' \in J, j \neq j'} d_{j'} [x_{j'}^L, x_{j'}^U] - \sum_{k \in K} q(p_{k,k}, [f_k^L, f_k^U], x_k) \right).$$

For a “quadratic” variable x_k , $k \in K$, one way to compute potentially tighter bounds is by solving the quadratic interval-equation

$$\begin{aligned} \sum_{j \in J} d_j [x_j^L, x_j^U] + \sum_{k' \in K, k' \neq k} q(p_{k',k'}, [e_{k'}, e_{k'}], x_{k'}) + \sum_{r \in K, r \neq k'} p_{k,r} [x_r^L, x_r^U], x_{k'} \\ + ([e_k, e_k] + \sum_{r \in K} (p_{k,r} + p_{r,k}) [x_r^L, x_r^U]) x_k + p_{k,k} x_k^2 \in [\ell, u]. \end{aligned}$$

However, since evaluating the argument of $q(\cdot)$ for each $k \in K$ may produce a huge computational overhead, especially for constraints with many bilinear terms, we compute the solution set of

$$\begin{aligned} \sum_{j \in J} d_j [x_j^L, x_j^U] + \sum_{\substack{k' \in K \\ k' \neq k}} \left(q(p_{k',k'}, [e_{k'}, e_{k'}], x_{k'}) + \sum_{\substack{r \in K \\ r \neq k'}} p_{k',r} [x_{k'}^L, x_{k'}^U] [x_r^L, x_r^U] \right) \\ + ([e_k, e_k] + \sum_{r \in K} (p_{k,r} + p_{r,k}) [x_r^L, x_r^U]) x_k + p_{k,k} x_k^2 \in [\ell, u], \quad (3.6) \end{aligned}$$

which can be performed more efficiently. If the intersection of the current domain $[x_k^L, x_k^U]$ of x_k with the solution set of (3.6) is empty, we can deduce infeasibility and prune the corresponding node. Otherwise, we may be able to tighten the bounds of x_k .

As in [16], all interval operations detailed in this section are performed in outward rounding mode.

3.4. Enforcement. In the enforcement call, a constraint handler has to check whether the current LP solution \tilde{x} is feasible for all its constraints. It can resolve an infeasibility by either adding cutting planes that separate \tilde{x} from the relaxation, by tightening bounds on a variable such that \tilde{x} is separated from the current domain, by pruning the current node from the branch-and-bound tree, or by performing a branching operation.

We have configured SCIP to call the enforcement method of the quadratic constraint handler with a lower priority than the enforcement method for the handler of integrality constraints. Thus, at the point where quadratic constraints are enforced, all integer variables take an integral value in the LP optimum \tilde{x} . For a violated quadratic constraint, we first perform a forward propagation step, see Section 3.3, which may prune the current node. If the forward propagation does not declare infeasibility, we call the separation method, see Section 3.2. If the separator fails to cut off \tilde{x} , we perform a spatial branching operation. We use the following branching rule to resolve infeasibility in a nonconvex quadratic constraint.

Branching rule. We consider each unfixed variable x_j that appears in a violated nonconvex quadratic constraint as a branching candidate. Let $x_j^l, x_j^u \in \overline{\mathbb{Q}}$ be the local lower and upper bounds of x_j , and $x_j^b \in (x_j^l, x_j^u)$ be the potential branching point for branching on x_j . Usually, we choose $x_j^b = \tilde{x}_j$. If, however, \tilde{x}_j is very close to one of the bounds, x_j^b is shifted inwards the interval. Thus, for $x_j^l, x_j^u \in \mathbb{Q}$, we let $x_j^b := \min\{\max\{\tilde{x}_j, \lambda x_j^l + (1 - \lambda)x_j^u\}, \lambda x_j^u + (1 - \lambda)x_j^l\}$, where the parameter λ is set to 0.2 in our experiments.

As suggested in [6], we select the branching variable w.r.t. its pseudocost values. The pseudocosts are used to estimate the objective change in the LP relaxation when branching downwards and upwards on a particular variable. The pseudocosts of a variable are defined as the average objective gains per unit change, taken over all nodes, where this variable has been chosen for branching, see [8] for details.

In classical pseudocost branching for integer variables, the distances of \tilde{x}_j to the nearest integers are used as multipliers of the pseudocosts. For continuous variables, we use another measure similar to “rb-int-br-rev” which was suggested in [6]: the distance of x_j^b to the bounds x_j^L and x_j^U for a variable x_j . This measure is motivated by the observation that the length of the domain determines the quality of the convexification. If the domain of x_j is unbounded, then the “convexification error of the variable x_j ” will be used as multiplicator. This value is computed by assigning to each variable the gap evaluated in \tilde{x} that is introduced by using a secant or McCormick underestimator for a nonconvex term that includes this variables.

As in [2], we combine the two estimates for downwards and upwards branching by multiplication rather than by a convex sum.

4. A constraint handler for Second-Order Cone constraints.

Constraints of the form

$$\sqrt{\sum_{i=1}^k \gamma + (\alpha_i(x_i + \beta_i))^2} \leq \alpha_0 y + \beta_0, \quad \alpha_0 y \geq -\beta_0, \quad (4.1)$$

where $\alpha_i, \beta_i \in \mathbb{Q}$, $i = 0, \dots, k$, $\gamma \in \mathbb{Q}_+$ are handled by a constraint handler for second-order cone constraints. Note that SOC constraints are convex, i.e., the nonlinear function on the left hand side of (4.1) is convex. Therefore, unlike nonconvex quadratic constraints, SOC constraints can be enforced by separation routines solely. First, the inequality $\alpha_0 y \geq -\beta_0$ is ensured by tightening the bounds of y accordingly. Next, if the current LP solution (\tilde{x}, \tilde{y}) violates some SOC constraint (4.1), then we add the valid gradient-based inequality

$$\eta + \frac{1}{\eta} \sum_{i=1}^k \alpha_i^2 (\tilde{x}_i + \beta_i)(x_i - \tilde{x}_i) \leq \alpha_0 y + \beta_0,$$

where $\eta := \sqrt{\sum_{i=1}^k \gamma + (\alpha_i(\tilde{x}_i + \beta_i))^2}$. Note that since (\tilde{x}, \tilde{y}) violates (4.1), one has $\eta > \alpha_0 \tilde{y} + \beta_0 \geq 0$. For the initial linear relaxation, no inequalities are added.

We also experimented with adding a linear outer-approximation as suggested in [7] a priori, but did not observe computational benefits. Thus, this option has been disabled for the experiments in Section 6.

5. Primal heuristics. When solving MIQCPs, we still make use of all default MILP primal heuristics of **SCIP**. Most of these heuristics aim at finding good integer and LP feasible solutions starting from the optimum of the LP relaxation. For details and a computational study of the primal MILP heuristics available in **SCIP**, see [9].

So far, we have implemented two additional primal heuristics for solving MIQCPs in **SCIP**, both of which are based on *large neighborhood search*.

QCP local search. There are several cases, where the MILP primal heuristics already yield feasible solutions for the MIQCP. However, the heuristics usually construct a point \hat{x} which is feasible for the MILP relaxation, i.e., the LP relaxation plus the integrality requirements, but violates some of the quadratic constraints. Such a point may, nevertheless, provide useful information, since it can serve as starting point for a local search.

The local search we currently use considers the space of continuous variables, i.e., if there are continuous variables in a quadratic part of a constraint, we solve a QCP obtained from the MIQCP by fixing all integer variables to the values of \hat{x} , using \hat{x} as starting point for the QCP solver. Each feasible solution of this QCP also is a feasible solution of the MIQCP.

RENS. Furthermore, we implemented an extended form of the relaxation enforced neighborhood search (RENS) heuristic [10]. This heuristic creates a sub-MIQCP problem by exploiting the optimal solution \tilde{x} of the LP relaxation at some node of the branch-and-bound-tree. In particular, it fixes all integer variables which take an integral value in \tilde{x} and restricts the bounds of all integer variables with fractional LP solution value to the two nearest integral values. This, hopefully much easier, sub-MIQCP is then partially solved by a separate **SCIP** instance. Obviously, each feasible solution of the sub-MIQCP is a feasible solution of the original MIQCP.

Note that, during the solution process of the sub-MIQCP, the QCP local search heuristic may be used along with the default **SCIP** heuristics. For some instances this works particularly well since, amongst others, RENS performs additional presolving reductions on the sub-MIQCP – which yields a better performance of the QCP solver.

6. Computational experiments. We conducted numerical experiments on three different test sets. The first is a test set of *mixed integer quadratic programs (MIQPs)* [22], i.e., problems with a quadratic objective function and linear constraints. Secondly, we selected a test set of *mixed integer conic programs (MICPs)* [27], which have been formulated as MIQCP. Finally, we assembled a test set of 24 general MIQCPs from the MINLPLib [14] and six constrained layout problems (`clay*`) from [15].

We will refer to these test sets as MIQP, MICP, and MINLP test sets. In Tables 1–3, each entry shows the number of seconds a certain solver needs to solve a problem. If the problem was not solved within the given time limit, the lower and upper bounds at termination are given. For

each instance, the fastest solution time or – in case all solvers hit the time limit – the best bounds, are depicted in bold face. Further, for each solver we calculated the geometric mean of the solution time (in which unsolved instances are accounted for with the time limit), and collected statistics on how often a solver solved a problem, computed the best dual bound, found the best primal solution value, or was the fastest among all solvers.

For our benchmark, we ran **SCIP** 1.2.0.7 using **CPLEX** 11.2.1 [19] as LP solver, **Ipopt** 3.8 [28] as QCP solver for the heuristics (cf. Section 5), and **LAPACK** 3.1.0 to compute eigenvalues. For comparison, we ran **BARON** 9.0.2 [26], **Couenne** 0.3 [6], **CPLEX** 12.1, **LindoGlobal** 6.0.1 [20], and **MOSEK** 6.0.0.55 [23]. Note that **BARON**, **Couenne**, and **LindoGlobal** can also be applied to general MINLPs. All solvers were run with a time limit of one hour, a final gap tolerance of 10^{-4} , and a feasibility tolerance of 10^{-6} on a 2.5 GHz Intel Core2 Duo CPU with 4 GB RAM and 6 MB Cache.

Mixed Integer Quadratic Programs. Table 6 presents the 25 instances from the MIQP test set [22]. We observe that due to the reformulation (3.1), 15 instances could be reformulated as mixed integer linear programs in the presolving state.

Table 1 compares the performance of **SCIP**, **BARON**, **Couenne**, and **CPLEX** on the MIQP test set. We did not run **LindoGlobal** since many of the MIQP instances exceed limitations of our **LindoGlobal** license. Note that some of the instances are nonconvex before applying the reformulation described in Section 3.1, so that we did not apply solvers which have only been designed for convex problems. Instance **ivalues** is the only instance that cannot be handled by **CPLEX** due to nonconvexity. Altogether, **SCIP** performs much better than **BARON** and **Couenne** and slightly better than **CPLEX** w.r.t. the mean computation time.

Mixed Integer Conic Programs. The MICP test set consists of three types of optimization problems, see Table 5. The **classical_XXX_YY** instances contain one convex quadratic constraint of the form $\sum_{j=1}^k x_j^2 \leq u$ for some $u \in \mathbb{Q}$, where **XXX** stand for the dimension k and **YY** is a problem index. The **robust_XXX_YY** instances contain one convex quadratic and one SOC constraint of dimension k . The **shortfall_XXX_YY** instances contain two SOC constraints of dimension k .

Table 2 compares the performance of **BARON**, **Couenne**, **CPLEX**, **MOSEK**, **LindoGlobal**, and **SCIP** on the MICP test set. We observe that on this specific test set **SCIP** outperforms **BARON**, **Couenne**, and **LindoGlobal**. It solves one instance more but is about 20% slower than the commercial solvers **CPLEX** and **MOSEK**.

Mixed Integer Quadratically Constrained Programs. The instances **lop97ic**, **lop97icx**, **pb302035**, **pb351535**, **qap**, and **qapw** were transformed into MILPs by presolving – which is due to the reformulation (3.1). The instances **nuclear***, **space25**, **space25a**, and **waste** are

TABLE 1

Results on MIQP instances. Each entry shows the number of seconds to solve a problem, or the bounds obtained after the one hour limit.

instance	BARON	Couenne	CPLEX	SCIP
iair04	$[-\infty, \infty]$	fail	37.52	228.27
iair05	$[-\infty, \infty]$	[25886, ∞)	30.71	113.65
ibc1	[1.792, 3.72]	[1.696, 3.98]	895.54	43.06
ibell3a	58.95	198.90	3.96	14.59
ibienst1	1048.04	[$-\infty$, 49.11]	2836.05	31.53
icap6000	[-2448496, -2441852]	fail	6.28	6.10
icvxqp1	[324603, 613559]	fail	[327522, 410439]	[0, 4451398]
ieild76	[729.5, 1081]	[808.3, 898.5]	13.50	41.28
ilaser0	$[-\infty, \infty]$	fail	[2409925, 2412734]	fail
imas284	[89193, 92241]	3139.12	4.36	27.33
imisc07	[2432, 2814]	[1696, 3050]	70.02	30.52
imod011	[-3.823, -3.843]	fail	[$-\infty, \infty$]	[$-\infty, 0$]
inug06-3rd	[177.8, 1434]	fail	[527.2, 1434]	[1152, ∞)
inug08	[1451, 14696]	[683.1, ∞)	2126.68	24.83
iportfolio	$[-\infty, 0]$	[-0.4944, ∞)	[-0.4944, -0.4937]	[-0.5251, 0]
iqap10	$[-\infty, \infty]$	[329.8, ∞)	1411.26	657.71
iqui	[-402.5, -108.6]	[-357.5, -126.3]	91.77	64.53
iran13x13	[2930, 3355]	[3014, 3476]	20.02	68.44
iran8x32	[5013, 5454]	[5034, 5629]	25.24	8.31
isqp0	$[-\infty, \infty]$	[- ∞ , -20137]	[-20338, -20320]	[$-\infty$, -19895]
isqp1	$[-\infty, \infty]$	[- ∞ , -18801]	[-19028, -18993]	[$-\infty$, -17883]
isqp	$[-\infty, \infty]$	[- ∞ , -20722]	[-21071, -21001]	[$-\infty, \infty$)
iswath2	[335.6, 661.9]	[335.9, 411.8]	212.15	121.29
itointqor	$[-\infty, -1146]$	fail	[-1150, -1147]	[$-\infty, 0$]
ivvalues	[-12.88, -0.4168]	[-6.054, -1.056]	-	[-172.6, ∞)
mean time	2923.78	3193.55	423.387	303.013
#solved	2	2	15	15
#best dual bound	4	5	21	16
#best primal sol.	5	3	23	15
#fastest	0	0	6	9

particularly difficult since they contain continuous variables that appear in quadratic terms with at least one bound at infinity. This prohibits to use the reformulation (3.1) for products of binary variables with a linear term. Further, generating secant and McCormick cuts for nonconvex terms is not possible. Thus, if the propagation algorithm cannot reduce domains for such unbounded variables, it may require many branching operations until reasonable variable bounds and a lower bound can be computed.

Table 3 compares the performance of BARON, Couenne, LindoGlobal, and SCIP on the MINLP test set. Figure 2 shows a performance profile for this particular test set. Regarding the number of solved instances, LindoGlobal performs best: it could solve two instances more than BARON and SCIP, which both solved six instances more than Couenne. SCIP was, however, significantly faster than the other solvers.

BARON wrongly declared the instance `product` to be infeasible and hit the time limit while parsing the instances `pb302035` and `pb351535`. Couenne wrongly declared the instances `product` and `waste` to be infeasible. Using a time limit of 3600 seconds, LindoGlobal and Couenne did not stop

Results on MICP instances. Each entry shows the number of seconds to solve a problem, or the bounds obtained after the one hour limit.

instance	BARON	Couenne	CPLEX	LindoGlobal	MOSEK	SCIP
classical_40_-0	207.24	178.72	1.60	38.25	12.79	33.16
classical_40_-1	7.09	114.49	1.01	217.55	22.28	5.53
classical_50_-0	[-0.091911, -0.09074]	[-0.09447, -0.09054]	41.33	[-0.09572, -0.09074]	161.17	2664.58
classical_50_-1	250.46	[-0.09595, -0.09459]	7.38	[-0.09593, -0.09476]	30.62	210.65
classical_200_-0	[-0.1247, -0.1077]	[-0.1255, -0.0951]	[-0.1231, -0.1106]	[-0.1256, -0.08574]	[-0.124, -0.1103]	[-0.1284, -0.108]
classical_200_-1	[-0.1269, -0.1149]	[-0.1283, -0.1036]	[-0.1257, -0.1164]	[-0.1284, -0.1093]	[-0.1266, -0.1162]	[-0.1311, -0.1162]
robust_-40_0	3473.03	[-0.09706, -0.07602]	0.67	3600.95	1.37	4.03
robust_-40_-1	1752.70	[-0.116, -0.07646]	0.64	249.72	2.88	2.29
robust_-50_0	[-0.08615, -0.08609]	[-0.1263, -0.0861]	1.88	165.45	0.90	1.51
robust_-50_-1	[-0.08574, -0.08569]	[-0.1274, -0.0857]	2.32	506.36	3.18	8.71
robust_-100_0	[-0.1013, -0.0932]	[-0.1514, -0.09747]	[-0.1043, -0.09724]	[-0.1542, -0.08833]	1210.86	1392.47
robust_-100_-1	[-0.07501, -0.0703]	[-0.1257, -0.0677]	525.14	[-0.1269, 0]	292.21	592.92
robust_-200_0	[-0.1722, -0.1363]	[-0.145, -0.1411]	fail	[-1, 0]	[-0.1468, -0.1411]	[-0.1452, -0.1411]
robust_-200_-1	[-0.1477, -0.1424]	[-0.1995, -0.1377]	[-0.1454, -0.1425]	[-1, 0]	[-0.1456, -0.1427]	[-0.1467, -0.1413]
shortfall_-40_0	102.17	333.76	247.99	1027.02	45.71	15.39
shortfall_-40_-1	5.99	133.49	5.71	288.12	13.72	3.00
shortfall_-50_0	[-1.098, -1.095]	[-1.102, -1.095]	1913.00	[-1.104, -1.095]	405.93	1602.81
shortfall_-50_-1	91.84	[-1.103, -1.099]	13.13	[-1.104, -1.102]	21.73	15.44
shortfall_-100_0	[-1.12, -1.114]	[-1.132, -1.102]	[-1.125, -1.104]	[-1.116, -1.114]	[-1.121, -1.114]	[-1.121, -1.114]
shortfall_-100_-1	[-1.109, -1.106]	[-1.113, -1.091]	3301.75	[-1.113, -1.105]	[-1.111, -1.106]	2152.56
shortfall_-200_0	[-1.149, -1.12]	[-1.146, -1.125]	[-1.1479, -1.08]	[-1.146, -1.126]	[-1.149, -1.119]	[-1.149, -1.119]
shortfall_-200_-1	[-1.115, -1.131]	[-1.152, -1.101]	[-1.1361, -1.089]	[-1.151, -1.135]	[-1.148, -1.134]	[-1.148, -1.134]
mean time	1202.41	2092.28	226.932	1552.67	228.392	288.956
#solved	8	4	14	8	14	15
#best dual bound	8	4	19	8	16	16
#best primal sol.	13	9	17	12	19	17
#fastest	0	0	8	0	4	3

TABLE 3

Results on MINLP instances. Each entry shows the number of seconds to solve a problem, or the bounds obtained after the one hour limit.

instance	BARON	Couenne	LindoGlobal	SCIP
clay0203m	1.56	2.03	43.13	0.15
clay0204m	48.25	4.79	85.50	0.52
clay0205m	971.83	27.73	1162.00	6.49
clay0303m	1.29	[$-\infty$, 29911]	62.17	0.37
clay0304m	14.77	16.31	187.38	0.90
clay0305m	3584.73	27.65	1112.42	7.45
du-opt	137.39	[$-\infty$, 8727, ∞]	2204.70	1.07
du-opt5	150.54	[$-\infty$, 2437, 9.012]	697.10	0.47
lop97ic	[2549, ∞]	[3826 , ∞]	[$-\infty$, ∞]	[3069, 4547]
lop97icx	[2812, 4415]	[3903 , 4272]	[0, 5259]	[3763, 4099]
nous1	641.19	[1.345, 1.567]	41.44	[1.195, 1.567]
nous2	0.97	2.69	0.36	1349.72
nuclear14a	[-12.26 , ∞]	[-12.26 , ∞]	[$-\infty$, ∞]	[$-\infty$, -1.105]
nuclear14b	[-2.078 , -1.107]	[$-\infty$, -2.234, ∞]	[$-\infty$, ∞]	[$-\infty$, -1.118]
nuclear14	[$-\infty$, ∞]	[$-\infty$, -1.12]	[$-\infty$, -1.126]	[$-\infty$, -1.122]
nucleararva	[$-\infty$, ∞]	[$-\infty$, -1.005]	[$-\infty$, ∞]	[$-\infty$, ∞]
nvs19	12.14	778.31	457.04	0.21
nvs23	44.54	[$-\infty$, -1109]	2533.14	0.40
pb302035	[$-\infty$, ∞]	fail	[622588, ∞]	[1138613 , 4019210]
pb351535	[$-\infty$, ∞]	fail	fail	[1710093 , 4976433]
product	fail	fail	[$-\infty$, -2092]	326.76
qap	[103040 , 388250]	[0, ∞]	[$-\infty$, ∞]	[24761, 410450]
qapw	[265372 , 391210]	[0, ∞]	[0, 405354]	[32191, 400150]
space25a	[99.99, 490.2]	[35.09, ∞]	[330.6 , 489.2]	[72.46, ∞]
space25	[84.91 , 520.9]	[42.68, ∞]	[33.07, 638.8]	[72.46, ∞]
tln12	[32.73, ∞]	[16.19, ∞]	[85.8 , 139.1]	[16.41, 91.6]
tln5	798.56	[6.592, 10.3]	174.02	32.56
tln6	[13.75, 15.3]	[7.801, 15.3]	182.23	[10.21, 15.3]
tln7	[12.38, 15.6]	[5.038, 16.1]	[14.2 , 15.6]	[7.016, 15]
waste	[306.7, 712.3]	fail	1532.71	[306.7, 670.6]
mean time	755.466	1215.11	995.697	400.464
#solved	13	7	15	13
#best dual bound	18	10	18	15
#best primal sol.	17	11	17	23
#fastest	0	0	4	12

after 4000 seconds for pb351535 and for pb302035, pb351535, respectively. Further, no bounds were reported in the log file.

CPLEX can be applied to 11 instances of this test set. The clay* and du-opt* instances were solved within seconds; 4 times CPLEX was fastest, 4 times SCIP was fastest. For the instances pb302035, pb351535, and qap, CPLEX found good primal solutions, but very weak lower bounds.

Evaluation of implemented MIQCP techniques. In order to evaluate the computational effects of the implemented techniques, we compare the default settings of SCIP with a series of settings where a single technique has been turned off at a time. The methods we evaluated are the reformulation (3.1) for products that involve binary variables, cf. Section 3.1, the handling of SOC constraints by the SOC constraint handler, cf. Section 4, the domain propagation for quadratic constraints during branch-and-bound, cf. Section 3.3, the detection of convexity for multi-

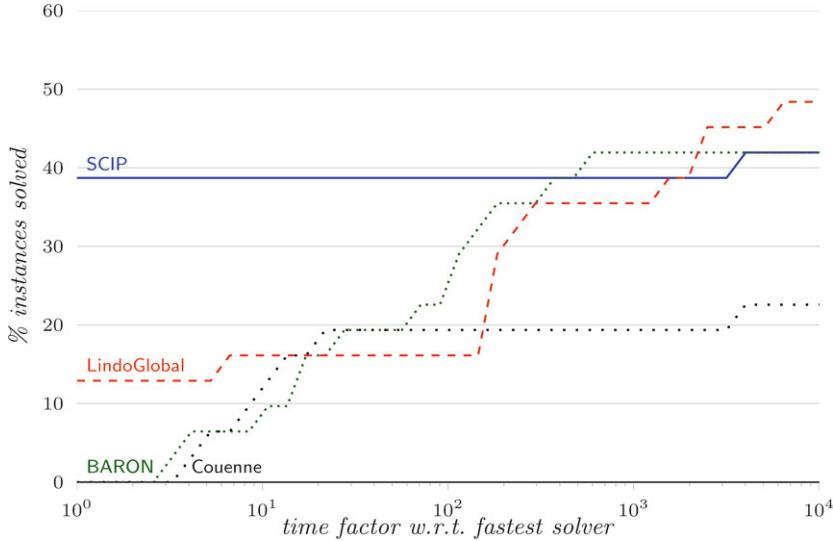


FIG. 2. Performance profile for MINLP test set.

variate quadratic functions, cf. Section 3.1, the QCP local search heuristic, cf. Section 5, and the extended RENS heuristic, cf. Section 5.

For each method, we evaluate the performance only on those instances from the test sets MIQP, MICP, and MINLP where the method to evaluate may have an effect (e.g., disabling the reformulation (3.1) is only evaluated on instances where this reformulation can be applied). The results are summarized in Table 4. For a number of performance measures we report the relative change caused by disabling a particular method.

We observe that deactivating one of the methods always leads to more deteriorations than improvements for both, the dual and primal bounds at termination. Except for one instance in the case of switching off binary reformulations, the number of solved instances remains equal or decreases.

Recognizing SOC constraints and convexity allows to solve instances of those special types much faster. Disabling domain propagation or one of the primal heuristics yields a small improvement w.r.t. computation time for easy instances, but results in weaker bounds for those instances which could not be solved within the time limit. We further observed that switching off the QCP local search heuristic increases the time until the first feasible solution is found by 93% and the time until the optimal solution is found by 26%. For RENS, the numbers are 12% and 43%, accordingly. Therefore, we still consider applying these techniques to be worthwhile.

7. Conclusions. In this paper, we have shown how a framework for constraint integer programming can be extended towards a solver for general MIQCPs. We implemented methods to correctly handle the quadratic

TABLE 4

Relative impact of implemented MIQCP methods. Percentages in columns 3–9 are relative to the size of the test set. Percentage in mean time column is relative to the mean time of SCIP with default settings.

disabled feature	size	solved	primal sol.		dual bound		running time		
			better	worse	better	worse	better	worse	mean
binary reform.	32	+3%	13%	22%	6%	25%	22%	34%	+3%
SOC upgrade	16	-69%	0%	69%	0%	100%	0%	69%	+1317%
domain prop.	48	±0%	4%	8%	6%	17%	29%	15%	-4%
convexity check	10	-20%	20%	20%	0%	30%	10%	30%	+159%
QCP local search	48	±0%	2%	17%	2%	4%	38%	17%	-6%
RENS heuristic	56	±0%	5%	9%	7%	7%	41%	11%	-3%

constraints. In order to speed up computations we further implemented MIQCP specific presolving, propagation, and separation methods. Furthermore, we discussed two large neighborhood search heuristics for MIQCP. The computational results indicate that this already suffices for building a solver which is competitive to state-of-the-art solvers like CPLEX, BARON, Couenne, and LindoGlobal. SCIP performed particularly well on the MIQP and MICP test sets, which contain a large number of linear constraints and a few quadratic constraints. These results meet our expectations, since SCIP already features several sophisticated MILP technologies.

We conclude that the extension of a full-scale MILP solver for handling MIQCP is a promising approach. The next step towards a full-scale MIQCP solver will be the incorporation of further MIQCP specific components into SCIP, e.g., more sophisticated separation routines [5, 24] and specialized constraint handlers, e.g., for bilinear covering constraints [25].

Acknowledgments. We like to thank Ambros M. Gleixner and Marc E. Pfetsch for their valuable comments on the paper. We further thank the anonymous reviewers for their constructive suggestions.

REFERENCES

- [1] K. ABHISHEK, S. LEYFFER, AND J. LINDEROTH, *Filmint: An outer-approximation-based solver for nonlinear mixed integer programs*, INFORMS Journal on Computing, **22** (2010), pp. 555–567.
- [2] T. ACHTERBERG, *Constraint Integer Programming*, PhD thesis, Technische Universität Berlin, 2007.
- [3] ———, *SCIP: Solving Constraint Integer Programs*, Math. Program. Comput., **1** (2009), pp. 1–41.
- [4] T. ACHTERBERG, T. BERTHOLD, T. KOCH, AND K. WOLTER, *Constraint integer programming: A new approach to integrate CP and MIP*, in Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008, L. Perron and M. Trick, eds., Vol. **5015** of LNCS, Springer, 2008, pp. 6–20.

- [5] X. BAO, N. SAHINIDIS, AND M. TAWARMALANI, *Multiterm polyhedral relaxations for nonconvex, quadratically-constrained quadratic programs*, Optimization Methods and Software, **24** (2009), pp. 485–504.
- [6] P. BELOTTI, J. LEE, L. LIBERTI, F. MARGOT, AND A. WÄCHTER, *Branching and bounds tightening techniques for non-convex MINLP*, Optimization Methods and Software, **24** (2009), pp. 597–634.
- [7] A. BEN-TAL AND A. NEMIROVSKI, *On polyhedral approximations of the second-order cone*, Math. Oper. Res., **26** (2001), pp. 193–205.
- [8] M. BÉNICHOU, J. M. GAUTHIER, P. GIRODET, G. HENTGES, G. RIBIÈRE, AND O. VINCENT, *Experiments in mixed-integer linear programming*, Math. Program., **1** (1971), pp. 76–94.
- [9] T. BERTHOLD, *Primal heuristics for mixed integer programs*. Diploma thesis, Technische Universität Berlin, 2006.
- [10] ———, *RENS – relaxation enforced neighborhood search*, ZIB-Report 07-28, Zuse Institute Berlin, 2007.
- [11] T. BERTHOLD, S. HEINZ, AND M.E. PFETSCH, *Nonlinear pseudo-boolean optimization: relaxation or propagation?*, in Theory and Applications of Satisfiability Testing – SAT 2009, O. Kullmann, ed., no. 5584 in LNCS, Springer, 2009, pp. 441–446.
- [12] R.E. BIXBY, M. FENELON, Z. GU, E. ROTHBERG, AND R. WUNDERLING, *MIP: theory and practice – closing the gap*, in System Modelling and Optimization: Methods, Theory and Applications, M. Powell and S. Scholtes, eds., Kluwer, 2000, pp. 19–50.
- [13] P. BONAMI, L.T. BIEGLER, A.R. CONN, G. CORNUÉJOLS, I.E. GROSSMANN, C.D. LAIRD, J. LEE, A. LODI, F. MARGOT, N.W. SAWAYA, AND A. WÄCHTER, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optim., **5** (2008), pp. 186–204.
- [14] M.R. BUSSIECK, A.S. DRUD, AND A. MEERAUS, *MINLPLib - a collection of test models for mixed-integer nonlinear programming*, INFORMS J. Comput., **15** (2003), pp. 114–119.
- [15] CMU-IBM MINLP PROJECT. <http://egon.cheme.cmu.edu/ibm/page.htm>.
- [16] F. DOMES AND A. NEUMAIER, *Quadratic constraint propagation*, Constraints, **15** (2010), pp. 404–429.
- [17] J.N. HOOKER, *Integrated Methods for Optimization*, International Series in Operations Research & Management Science, Springer, New York, 2007.
- [18] R. HORST AND H. TUY, *Global Optimization: Deterministic Approaches*, Springer, 1990.
- [19] IBM, *CPLEX*. <http://ibm.com/software/integration/optimization/cplex>.
- [20] Y. LIN AND L. SCHRAGE, *The global solver in the LINDO API*, Optimization Methods and Software, **24** (2009), pp. 657–668.
- [21] G. MCCORMICK, *Computability of global solutions to factorable nonconvex programs: Part I-Convex Underestimating Problems*, Math. Program., **10** (1976), pp. 147–175.
- [22] H. MITTELMANN, *MIQP test instances*. <http://plato.asu.edu/ftp/miqp.html>.
- [23] MOSEK CORPORATION, *The MOSEK optimization tools manual*, 6.0 ed., 2009.
- [24] A. SAXENA, P. BONAMI, AND J. LEE, *Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations*, Tech. Rep. RC24695, IBM Research, 2008. to appear in Math. Program.
- [25] M. TAWARMALANI, J.-P.P. RICHARD, AND K. CHUNG, *Strong valid inequalities for orthogonal disjunctions and bilinear covering sets*, Math. Program., **124** (2010), pp. 481–512.
- [26] M. TAWARMALANI AND N. SAHINIDIS, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, Kluwer Academic Publishers, 2002.

- [27] J.P. VIELMA, S. AHMED, AND G.L. NEMHAUSER, *A lifted linear programming branch-and-bound algorithm for mixed integer conic quadratic programs*, INFORMS J. Comput., **20** (2008), pp. 438–450.
- [28] A. WÄCHTER AND L.T. BIEGLER, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Math. Program., **106** (2006), pp. 25–57.

APPENDIX

In this section, detailed problem statistics are presented for the three test sets, MICP ([Table 5](#)), MIQP, and MINLP (both in [Table 6](#)). The columns belonging to “original problem” state the structure of the original problem. The “presolved problem” columns show statistics about the MIQCP after SCIP has applied its presolving routines, including the ones described in Section 3.1. The columns “vars”, “int”, and “bin” show the number of all variables, the number of integer variables, and the number of binary variables, respectively. The columns “linear”, “quad”, and “soc” show the number of linear, quadratic, and second-order cone constraints, respectively. The column “conv” indicates whether all quadratic constraints of the presolved MIQCP are convex or whether at least one of them is nonconvex.

TABLE 5
Statistics of instances in MICP test set.

instance	original problem				presolved problem						
	vars	int	bin	linear	quad	vars	int	bin	linear	quad	soc
classical_40_0	120	0	40	82	1	120	0	40	82	1	0
classical_40_1	120	0	40	82	1	120	0	40	82	1	0
classical_50_0	150	0	50	102	1	150	0	50	102	1	0
classical_50_1	150	0	50	102	1	150	0	50	102	1	0
classical_200_0	600	0	200	402	1	600	0	200	402	1	0
classical_200_1	600	0	200	402	1	600	0	200	402	1	0
robust_40_0	163	0	41	124	2	163	0	41	124	1	1
robust_40_1	163	0	41	124	2	163	0	41	124	1	1
robust_50_0	203	0	51	154	2	203	0	51	154	1	1
robust_50_1	203	0	51	154	2	203	0	51	154	1	1
robust_100_0	403	0	101	304	2	403	0	101	304	1	1
robust_100_1	403	0	101	304	2	403	0	101	304	1	1
robust_200_0	803	0	201	604	2	803	0	201	604	1	1
robust_200_1	803	0	201	604	2	803	0	201	604	1	1
shortfall_40_0	164	0	41	125	2	164	0	41	125	0	2
shortfall_40_1	164	0	41	125	2	164	0	41	125	0	2
shortfall_50_0	204	0	51	155	2	204	0	51	155	0	2
shortfall_50_1	204	0	51	155	2	204	0	51	155	0	2
shortfall_100_0	404	0	101	305	2	404	0	101	305	0	2
shortfall_100_1	404	0	101	305	2	404	0	101	305	0	2
shortfall_200_0	804	0	201	605	2	804	0	201	605	0	2
shortfall_200_1	804	0	201	605	2	804	0	201	605	0	2

TABLE 6
Statistics of instances in MIQP (first part) and MINLP (second part) test set.

instance	original problem					presolved problem					conv
	vars	int	bin	linear	quad	vars	int	bin	linear	quad	
iair04	8905	0	8904	823	1	12848	0	7362	17464	0	✓
iair05	7196	0	7195	426	1	10574	0	6117	14218	0	✓
ibc1	1752	0	252	1913	1	866	0	252	1438	0	✓
ibell3a	123	29	31	104	1	129	29	31	161	1	✓
ibienst1	506	0	28	576	1	473	0	28	592	0	✓
icap6000	6001	0	6000	2171	1	7323	0	5865	6362	0	✓
icvxqp1	10001	10000	0	5000	1	10003	9998	2	5006	1	✓
ieild76	1899	0	1898	75	1	2685	0	1898	3168	0	✓
ilaser0	1003	151	0	2000	1	1003	151	0	1000	1	✓
imas284	152	0	150	68	1	228	0	150	299	0	✓
imisc07	261	0	259	212	1	360	0	238	598	0	✓
imod011	10958	1	96	4480	1	8963	1	96	2730	1	✓
inug06-3rd	2887	0	2886	3972	1	3709	0	2886	7779	0	✓
inug08	1633	0	1632	912	1	2217	0	1632	3076	0	✓
iportfolio	1201	192	775	201	1	1201	192	775	201	1	✓
iqap10	4151	0	4150	1820	1	5879	0	4150	9047	0	✓
iqiu	841	0	48	1192	1	871	0	48	1285	0	✓
iran13x13	339	0	169	195	1	468	0	169	585	0	✓
iran8x32	513	0	256	296	1	651	0	256	713	0	✓
isqp0	1001	50	0	249	1	1001	50	0	249	1	✓
isqp1	1001	0	100	249	1	1068	0	100	480	1	
isqp	1001	50	0	249	1	1001	50	0	249	1	✓
iswath2	6405	0	2213	483	1	8007	0	2213	5631	0	✓
itointqor	51	50	0	0	1	51	50	0	0	1	✓
ivalues	203	202	0	1	1	203	202	0	1	1	
clay0203m	30	0	18	30	24	27	0	15	27	24	✓
clay0204m	52	0	32	58	32	48	0	28	54	32	✓
clay0205m	80	0	50	95	40	75	0	45	90	40	✓
clay0303m	33	0	21	30	36	31	0	19	29	36	✓
clay0304m	56	0	36	58	48	54	0	34	57	48	✓
clay0305m	85	0	55	95	60	81	0	51	93	60	✓
du-opt	21	13	0	9	1	21	13	0	5	1	✓
du-opt5	21	13	0	9	1	19	11	0	4	1	✓
lop97ic	1754	831	831	52	40	5228	708	708	11521	0	✓
lop97icx	987	831	68	48	40	488	68	68	1138	0	✓
nous1	51	0	2	15	29	47	0	2	11	29	
nous2	51	0	2	15	29	47	0	2	11	29	
nvs19	9	8	0	0	9	9	8	0	0	9	
nvs23	10	9	0	0	10	10	9	0	0	10	
pb302035	601	0	600	50	1	1199	0	600	1847	0	✓
pb351535	526	0	525	50	1	1048	0	525	1619	0	✓
product	1553	0	107	1793	132	446	0	92	450	82	
qap	226	0	225	30	1	449	0	225	702	0	✓
qapw	451	0	225	255	1	675	0	225	930	0	✓
space25	893	0	750	210	25	767	0	716	118	25	
space25a	383	0	240	176	25	308	0	240	101	25	
nuclear14	1562	0	576	624	602	986	0	576	48	602	
nuclear14a	992	0	600	49	584	1568	0	600	2377	560	
nuclear14b	1568	0	600	1225	560	1568	0	600	1225	560	
nucleararva	351	0	168	50	267	327	0	144	24	267	
tln12	168	156	12	60	12	180	144	24	85	11	
tln5	35	30	5	25	5	35	30	5	20	5	
tln6	48	42	6	30	6	48	42	6	24	6	
tln7	63	56	7	35	7	63	56	7	28	7	
waste	2484	0	400	623	1368	1238	0	400	516	1230	

PART VII:

COMBINATORIAL OPTIMIZATION

COMPUTATION WITH POLYNOMIAL EQUATIONS AND INEQUALITIES ARISING IN COMBINATORIAL OPTIMIZATION

JESUS A. DE LOERA*, PETER N. MALKIN†, AND PABLO A. PARRILO‡

Abstract. This is a survey of a recent methodology to solve systems of polynomial equations and inequalities for problems arising in combinatorial optimization. The techniques we discuss use the algebra of multivariate polynomials with coefficients over a field to create large-scale linear algebra or semidefinite programming relaxations of many kinds of feasibility or optimization questions.

Key words. Polynomial equations and inequalities, combinatorial optimization, Nullstellensatz, Positivstellensatz, graph colorability, max-cut, stable sets, semidefinite programming, large-scale linear algebra, semi-algebraic sets, real algebra.

AMS(MOS) subject classifications. 90C27, 90C22, 68W05.

1. Introduction. A wide variety of problems in optimization can be easily modeled using *systems of polynomial equations and inequalities*. Feasibility and optimization problems translate, either directly or via branching, into the problem of finding a solution of a system of equations and inequalities. In this survey paper, we explain how to manipulate such systems for finding solutions or proving that they do not exist. Although these techniques work in general, we are particularly motivated by problems of combinatorial origin. For example, in the case of graphs, here is how one can think about stable sets, k -colorability and max-cut problems in terms of polynomial (non-linear) constraints:

PROPOSITION 1.1. *Let $G = (V, E)$ be a graph.*

- *For a given positive integer k , consider the following polynomial system:*

$$x_i^2 - x_i = 0 \quad \forall i \in V, \quad x_i x_j = 0 \quad \forall (i, j) \in E \quad \text{and} \quad \sum_{i \in V} x_i = k.$$

This system is feasible if and only if G has a stable set of size k .

*Department of Mathematics, University of California at Davis, Davis, CA 95616 (deloera@math.ucdavis.edu); partially supported by NSF DMS-0914107 and an IBM OCR award.

†Department of Mathematics, University of California at Davis, Davis, CA 95616 (malkin@math.ucdavis.edu); partially supported by an IBM OCR award.

‡Laboratory for Information and Decision Systems, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 (parrilo@mit.edu); partially supported by AFOSR MURI 2003-07688-1 and NSF FRG DMS-0757207.

- For a positive integer k , consider the following polynomial system of $|V| + |E|$ polynomials equations:

$$x_i^k - 1 = 0 \quad \forall i \in V \quad \text{and} \quad \sum_{s=0}^{k-1} x_i^{k-1-s} x_j^s = 0 \quad \forall (i, j) \in E.$$

The graph G is k -colorable if and only if this system has a complex solution. Furthermore, when k is odd, G is k -colorable if and only if this system has a common root over $\overline{\mathbb{F}}_2$, the algebraic closure of the finite field with two elements.

- We can represent the set of cuts of G (i.e., bipartitions on V) as the 0-1 incidence vectors

$$SG := \{\chi^F : F \subseteq E \text{ is contained in a cut of } G\} \subseteq \{0, 1\}^E.$$

Thus, the max cut problem with non-negative weights w_e on the edges $e \in E$ is

$$\max \left\{ \sum_{e \in E} w_e x_e : x \in SG \right\}.$$

The vectors χ^F are the solutions of the polynomial system

$$x_e^2 - x_e = 0 \quad \forall e \in E, \quad \text{and} \quad \prod_{i \in T} x_i = 0 \quad \forall T \text{ an odd cycle in } G.$$

There are many other combinatorial problems that can be modeled concisely by polynomial systems (see [9] and the many references therein). In fact, a given problem can often be modeled non-linearly in many different ways, and in practice choosing a “good” formulation is critical for an efficient solution.

Given a polynomial system encoding a combinatorial question, we explain how to use two famous algebraic identities to derive solution methods. In what follows, let \mathbb{K} denote a field and let $\mathbb{K}[x_1, \dots, x_n] = \mathbb{K}[x]$ denote the ring of polynomials in n variables with coefficients over \mathbb{K} . The situation is slightly different depending on whether only equations are being considered, or if there also inequalities (more precisely, on whether the underlying field \mathbb{K} is formally real):

1. First, suppose that the system contains only the polynomial equations $f_1(x) = 0, f_2(x) = 0, \dots, f_s(x) = 0$ where $f_1, \dots, f_s \in \mathbb{K}[x]$. We explain how to generate a finite sequence of *linear algebra* systems over \mathbb{K} which terminate with either a solution over $\overline{\mathbb{K}}$, the algebraic closure of \mathbb{K} , or provide a certificate of infeasibility. Crucially for practical computation, the linear algebra systems are over \mathbb{K} , not $\overline{\mathbb{K}}$. The calculations reduce to matrix manipulations over \mathbb{K} , mostly rank computations. The techniques we use are a

specialization of prior techniques from computational algebra (see [37, 20, 21, 38]). As it turns out this technique is particularly effective when the number of solutions is finite, when \mathbb{K} is a finite field and when the system has nice combinatorial information (see [9]).

2. Second, several authors (see e.g. [23, 41, 29] and references therein) have considered the solvability (over the reals) of systems of polynomial equations and inequalities. It was shown that in this situation there is a way to set up the feasibility problem

$$\exists x \in \mathbb{R}^n \text{ s.t. } f_1(x) = 0, \dots, f_s(x) = 0, g_1(x) \geq 0, \dots, g_k(x) \geq 0,$$

where $f_1, \dots, f_s, g_1, \dots, g_k \in \mathbb{R}[x]$, as a sequence of semidefinite programs terminating with a feasible solution (see [41, 29]). Once more, the combinatorial structure can help in the understanding of the structure of these relaxations, as is well-known from the case of stable sets [32] and max-cut [28]. In recent work, Gouveia et al. [15, 14] considered a sequence of semidefinite relaxations of the convex hull of real solutions of a polynomial system encoding a combinatorial problem. They called these approximations *theta bodies* because, for stable sets of graphs, the first theta body in this hierarchy is exactly Lovász's theta body of a graph [32].

The common central idea to both of the relaxations procedures described above is to use the right *infeasibility certificates* or *theorems of alternative*. Just as Farkas' lemma is a centerpiece for the development of Linear Programming, here the key point is that the infeasibility of polynomial systems can *always* be certified by particular algebraic identities (on non-linear polynomials). To find these infeasibility certificates we rely either on *linear algebra* or *semidefinite programming* (for a quick overview of semidefinite programming see [51]).

We now introduce some necessary notation and algebraic concepts. For a detailed introduction we recommend the books [2, 5, 6, 36]. In the paper \mathbb{K} denotes a field and when the distinction is necessary we denote its algebraic closure by $\overline{\mathbb{K}}$. Let $\mathbb{K}[x_1, \dots, x_n]$ denote the ring of polynomials in n variables with coefficients over \mathbb{K} , which will be abbreviated as $\mathbb{K}[x]$. We denote the monomials in the polynomial ring $\mathbb{K}[x]$ as $x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ for $\alpha \in \mathbb{N}^n$. The degree of x^α is $\deg(x^\alpha) := |\alpha| := \sum_{i=1}^n \alpha_i$. The degree of a polynomial $f = \sum_{\alpha \in \mathbb{N}^n} f_\alpha x^\alpha$, written $\deg(f)$, is the maximum degree of x^α where $f_\alpha \neq 0$ for $\alpha \in \mathbb{N}^n$. Given a set of polynomials $F \subset \mathbb{K}[x]$, we write $\deg(F)$ for the maximum degree of the polynomials in F . Given a set of polynomials $F := \{f_1, \dots, f_m\} \subseteq \mathbb{K}[x]$, we define the *ideal* generated by F as

$$\mathbf{ideal}(F) := \left\{ \sum_{i=1}^m \beta_i f_i \mid \beta_i \in \mathbb{K}[x] \right\}.$$

To study of solutions of a system over a non-algebraically closed field like \mathbb{R} requires extra structure. Given a set of real polynomials $G := \{g_1, \dots, g_m\} \subseteq \mathbb{R}[x]$, following page 86 in Section 4.2 of [2], we define the *cone* generated by G as

$$\mathbf{cone}(G) := \left\{ \sum_{\alpha \in \{0,1\}^n} s_\alpha g^\alpha \mid s_\alpha \in \mathbb{R}[x] \text{ is SOS} \right\}$$

where $g^\alpha := \prod_{i=1}^m g_i^{\alpha_i}$ and a polynomial $s(x) \in \mathbb{R}[x]$ is SOS if it can be written as a *sum of squares* of other polynomials, that is, $s(x) = \sum_i q_i^2(x)$ for some $q_i(x) \in \mathbb{R}[x]$. We note that the cone of G is also called a *preordering* generated by G in [36]. If $s(x)$ is SOS, then clearly $s(x) \geq 0$ for all $x \in \mathbb{R}^n$. The sum in the definition of $\mathbf{cone}(G)$ is finite, with a total of 2^m terms, corresponding to the subsets of $\{g_1, \dots, g_m\}$.

The notions of *ideal* and *cone* are standard in algebraic geometry, but they also have inherent convex geometry: Ideals are affine sets and cones are closed under convex combinations and non-negative scalings, i.e., they are actually cones in the convex geometry sense. Ideals and cones are used for deriving new *valid constraints*, which are logical consequences of the given constraints. For example, notice that by construction, every polynomial in $\mathbf{ideal}(\{f_1, \dots, f_m\})$ vanishes in the solution set of the system $f_1(x) = 0, \dots, f_m(x) = 0$ over the algebraic closure of \mathbb{K} . Similarly, every element of $\mathbf{cone}(\{g_1, \dots, g_m\})$ is clearly non-negative on the feasible set of $g_1(x) \geq 0, \dots, g_m(x) \geq 0$ over \mathbb{R} .

It is well-known that optimization algorithms are intimately tied to the development of infeasibility certificates. For example, the simplex method is closely related to Farkas' lemma. Our starting point is a generalization of this famous principle. We start with a description of two powerful infeasibility certificates for polynomial systems which generalize the classical ones for linear optimization. First, as motivation, recall from elementary linear algebra the “Fredholm alternative theorem” (e.g., see page 28 Corollary 3.1.b in [46]):

THEOREM 1.1 (Fredholm's alternative). *Given a matrix $A \in \mathbb{K}^{m \times n}$ and a vector $b \in \mathbb{K}^m$,*

$$\nexists x \in \mathbb{K}^n \text{ s.t. } Ax + b = 0 \Leftrightarrow \exists \mu \in \mathbb{K}^m \text{ s.t. } \mu^T A = 0, \mu^T b = 1.$$

It turns out that there are much stronger versions for general polynomials, which unfortunately do not seem to be widely known among optimizers (for more details see e.g., [5]).

THEOREM 1.2 (Hilbert's Nullstellensatz). *Let $F := \{f_1, \dots, f_m\} \subseteq \mathbb{K}[x]$. Then,*

$$\nexists x \in \overline{\mathbb{K}}^n \text{ s.t. } f_1(x) = 0, \dots, f_s(x) = 0 \Leftrightarrow 1 \in \mathbf{ideal}(F).$$

Note that $1 \in \mathbf{ideal}(F)$ means that there exist polynomials $\beta_1, \dots, \beta_m \in \mathbb{K}[x]$ such that $1 = \sum_{i=1}^m \beta_i f_i$, and this polynomial identity is thus a *certificate of infeasibility*. Fredholm's alternative theorem is simply a linear version of Hilbert's Nullstellensatz where all the polynomials are linear and the β_i 's are constant.

EXAMPLE 1. Consider the following set of polynomials in $\mathbb{R}[x_1, x_2, x_3]$:

$$F := \{f_1 := x_1^2 - 1, f_2 := 2x_1x_2 + x_3, f_3 := x_1 + x_2, f_4 := x_1 + x_3\}.$$

By the Nullstellensatz, the system $f_1(x) = 0, f_2(x) = 0, f_3(x) = 0, f_4(x) = 0$ is infeasible over \mathbb{C} if and only if there exist polynomials $\beta_1, \beta_2, \beta_3, \beta_4 \in \mathbb{R}[x_1, x_2, x_3]$ that satisfy the polynomial identity $\beta_1 f_1 + \beta_2 f_2 + \beta_3 f_3 + \beta_4 f_4 = 1$. Here, the system is infeasible, so there exist such polynomials as follows:

$$\beta_1 = -1 - \frac{2}{3}x_2, \beta_2 = -\frac{2}{3} + \frac{1}{3}x_1, \beta_3 = -\frac{2}{3} + \frac{4}{3}x_1, \beta_4 = \frac{2}{3} - \frac{1}{3}x_1.$$

The resulting identity provides a certificate of infeasibility of the system.

Now, the two theorems above deal only with the case of equations. The inclusion of inequalities in the problem formulation poses additional algebraic challenges because we need to take into account special properties of the reals. Consider first the case of linear inequalities, which is familiar to optimizers, where linear programming duality provides the following characterization:

THEOREM 1.3 (Farkas' lemma). Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^{k \times n}$, and $d \in \mathbb{R}^k$.

$$\begin{aligned} & \nexists x \in \mathbb{R}^n \text{ s.t. } Ax + b = 0, Cx + d \geq 0 \\ & \Updownarrow \\ & \exists \lambda \in \mathbb{R}_+^m, \exists \mu \in \mathbb{R}^k \text{ s.t. } \mu^T A + \lambda^T C = 0, \mu^T b + \lambda^T d = -1. \end{aligned}$$

Again, although not widely known in optimization, it turns out that similar certificates do exist for non-linear systems of polynomial equations and inequalities over the reals. The result essentially appears in this form in [2] and is due to Stengle [49].

THEOREM 1.4 (Positivstellensatz). Let $F := \{f_1, \dots, f_m\} \subset \mathbb{R}[x]$ and $G := \{g_1, \dots, g_k\} \subset \mathbb{R}[x]$.

$$\begin{aligned} & \nexists x \in \mathbb{R}^n \text{ s.t. } f_1(x) = 0, \dots, f_m(x) = 0, g_1(x) \geq 0, \dots, g_k(x) \geq 0 \\ & \Updownarrow \\ & \exists f \in \mathbf{ideal}(F), \exists g \in \mathbf{cone}(G) \text{ s.t. } f(x) + g(x) = -1. \end{aligned}$$

The theorem states that for every infeasible system of polynomial equations and inequalities, there exists a simple polynomial identity of the form $\sum_{i=1}^m \beta_i f_i + \sum_{\alpha \in \{0,1\}^n} s_\alpha g^\alpha = -1$ for some $\beta_i, s_\alpha \in \mathbb{R}[x]$ where s_α are SOS, that directly gives a certificate of infeasibility of real solutions.

EXAMPLE 2. Consider the polynomial system $\{f = 0, g \geq 0\}$, where

$$f := x_2 + x_1^2 + 2 = 0, \quad g := x_1 - x_2^2 + 3 \geq 0.$$

By the Positivstellensatz, there are no solutions $(x_1, x_2) \in \mathbb{R}^2$ if and only if there exist polynomials $\beta, s_1, s_2 \in \mathbb{R}[x_1, x_2]$ that satisfy

$$\beta \cdot f + s_1 + s_2 \cdot g = -1 \quad \text{where } s_1 \text{ and } s_2 \text{ are SOS.}$$

Here, the system is infeasible, so there exist such polynomials as follows:

$$s_1 = \frac{1}{3} + 2 \left(x_2 + \frac{3}{2} \right)^2 + 6 \left(x_1 - \frac{1}{6} \right)^2, s_2 = 2 \text{ and } \beta = -6.$$

The resulting identity provides a certificate of infeasibility of the system.

Of course, we are very concerned with the effective practical computation of the infeasibility certificates. For the sake of computation and complexity, we must worry about the growth of degrees and thus the growth in the encoding size of the infeasibility certificates. Here, we define the degree of a Nullstellensatz certificate $\sum_{i=1}^m \beta_i f_i = 1$ as $\max_i \{\deg(\beta_i f_i)\}$ and the degree of a Positivstellensatz certificate $\sum_{i=1}^m \beta_i f_i + \sum_{\alpha \in \{0,1\}^n} s_\alpha g^\alpha = -1$ as the larger of $\max_i \{\deg(\beta_i f_i)\}$ and $\max_\alpha \{\deg(s_\alpha g^\alpha)\}$. On the negative side, the degrees of the certificates are expected to grow at least linearly leading to exponential growth in the encoding size of the certificates simply because the NP-hardness of the original combinatorial questions; see e.g. [9]. At the same time, tight exponential upper bounds on the degrees have been derived (see e.g. [22], [16] and references therein). Nevertheless, for many problems of practical interest, it is often the case that it is possible to prove infeasibility using low-degree certificates (see [8, 7]). Even more important is the fact that for a fixed degree of the certificates, the calculations are polynomial time (see Lemma 2.1 and [41]) and can be reduced to either linear algebra or semidefinite programming. We summarize the strong analogies between the case of linear equations and inequalities with high-degree polynomial systems in the following table:

TABLE 1
Infeasibility certificates and their associated computational techniques.

Degree\Field	Arbitrary	Real
Linear	<i>Fredholm Alternative</i>	<i>Farkas' Lemma</i>
	Linear Algebra	Linear Programming
Polynomial	<i>Nullstellensatz</i>	<i>Positivstellensatz</i>
	Bounded degree Linear Algebra	Bounded degree SDP

It is important to remark that just as in the classical case of linear programming, the problem of computation of certificates has very natural primal-dual formulations, with the corresponding primal and dual variables playing distinct, but well-defined roles. For example, in the case of

Fredholm's alternative, the primal variables are the variables x_1, \dots, x_n while there is a dual variable for each equation. For Nullstellensatz and Positivstellensatz there is a similar duality, based on linear duality and semidefinite programming duality, respectively. In what follows, we use the most intuitive or convenient set-up and we leave to the reader the exercise of transferring the results to the corresponding dual version.

The remainder of the paper is divided in two main sections: Section 2 is a study of the Hilbert Nullstellensatz, for general fields, used in the solution of systems of equations. In Section 3, we survey the use of the Positivstellensatz in the context of solving systems of equations and inequalities over the reals. Both sections contain combinatorial applications that show why these techniques can be of interest in this setting. The focus of the combinatorial results is understanding those situations when a constant degree certificate is enough to show infeasibility. These are situations when hard combinatorial problems have polynomial time algorithms and as such provide structural insight. Finally, in Section 4, we describe a methodology, common to both approaches, to recover feasible solutions of the original combinatorial problem from the outcome of these relaxations. In addition, we have included an Appendix A that contains proofs of some the results used in the main body of the paper that are either hard to find or whose original proof, available elsewhere, is not written in the language of this survey.

To conclude the introduction we include some more notation and terminology. The *variety* of F over \mathbb{K} , written $\mathcal{V}_{\mathbb{K}}(F)$, is the set of common zeros of polynomials in F in \mathbb{K}^n , that is, $\mathcal{V}_{\mathbb{K}}(F) := \{v \in \mathbb{K}^n : f(v) = 0 \forall f \in I\}$. Also, $\mathcal{V}_{\overline{\mathbb{K}}}(F)$, the variety of F over $\overline{\mathbb{K}}$, is the set of common zeros of F in $\overline{\mathbb{K}}^n$. Note that in combinatorial problems, the *variety* of a polynomial system typically has finitely many solutions (e.g., colorings, cuts, stable sets, etc.). For an ideal $I \subseteq \mathbb{K}[x]$, when $\mathcal{V}_{\overline{\mathbb{K}}}(I)$ is finite, the ideal is called *zero-dimensional* (this is the case for all of the applications considered here). We say that a system of polynomial equations is a *combinatorial system* when its variety encodes a combinatorial problem (e.g., zeros represent stable sets, colorings, matchings, etc.) and it is zero-dimensional.

An ideal $I \subseteq \mathbb{K}[x]$ is *radical* if $f^k \in I$ for some positive integer k implies $f \in I$. We denote by \sqrt{I} the ideal of all polynomials $f \in \mathbb{K}[x]$ such that $f^k \in I$ for some positive integer k . The ideal \sqrt{I} is necessarily radical and it is called the radical ideal of I . Note that I is radical if and only if $I = \sqrt{I}$. Given a vector space W over a field \mathbb{K} , we write $\dim(W)$ for the dimension of W . Given vector spaces $U \subseteq W$, we write W/U as the vector space quotient. Recall that $\dim(W/U) = \dim(W) - \dim(U)$. Given a set $F \subset \mathbb{K}[x]$, $\text{span}(F)$ denotes the vector space generated by F over the field \mathbb{K} . Please note the distinction between the vector space $\text{span}(F)$ and the ideal $\text{ideal}(F)$.

2. Solving combinatorial systems of equations. In this section, we wish to solve a given system of polynomial equations $f_1(x) = 0, f_2(x) = 0, \dots, f_m(x) = 0$ where $f_1, \dots, f_m \in \mathbb{K}[x]$. The systems we consider have finitely many solutions, each corresponding to a combinatorial object. To simplify our arguments we also assume that \mathbb{K} is algebraically closed, i.e., $\mathbb{K} = \overline{\mathbb{K}}$. We abbreviate this system as $F(x) = 0$ where $F := \{f_1, \dots, f_m\} \subset \mathbb{K}[x]$. Here, by solving a system, we mean first determining if $F(x) = 0$ is feasible over \mathbb{K} , and furthermore finding a solution (or all solutions) of $F(x) = 0$ if feasible. The literature on polynomial solving is very extensive and it continues to be an area of active research (see [50, 6, 10] for an overview and background).

Here we choose to focus on techniques that fit well with traditional optimization methods. The main idea is that solving a polynomial system of equations can be reduced to solving a sequence of linear algebra problems. The foundations of this technique can be traced back to ([37, 20, 21, 38]). The specific approach we take to present this technique is closest to that of Mourrain in [37]. Variants of this technique have been applied to stable sets [9, 35], vertex coloring [8, 35], satisfiability (see e.g., [3]) and cryptography (see for example [4]). This technique is also strongly related to Border basis and Gröbner basis techniques, which can also be viewed in terms of linear algebra computations (see e.g., [20, 21, 38, 50]).

The linear algebra systems of equations have primal and dual representations in the sense of Fredholm's lemma. Specifically, in this survey, the primal approach solves a linear system to find constant multipliers $\mu \in \mathbb{K}^m$ such that $1 = \sum_{i=1}^m \mu_i f_i$ providing a certificate of (non-linear) infeasibility. Then, the dual approach aims to find a vector λ with entries in \mathbb{K} indexed by monomials such that $\sum_{\alpha} \lambda_{x^{\alpha}} f_{i,\alpha} = 0$ for all $i = 1, \dots, m$ and $\lambda_1 = 1$ where $f_i = \sum_{\alpha} f_{i,\alpha} x^{\alpha}$ for all i . As we see in Section 2.2, the dual approach amounts to constructing linear relaxations of the set of feasible solutions. In Sections 2.1 and 2.2, we present the primal and dual approaches respectively.

2.1. Linear algebra certificates. Consider the following corollary of Hilbert's Nullstellensatz: If there exist constants $\mu \in \mathbb{K}^m$ such that $\sum_{i=1}^m \mu_i f_i = 1$, then the polynomial system $F(x) = 0$ must be infeasible. In other words, if the system $F(x) = 0$ is infeasible, then $1 \in \text{span}(F)$. The crucial point here is that determining whether there exists a $\mu \in \mathbb{K}^m$ such that $\sum_{i=1}^m \mu_i f_i = 1$ is a linear algebra problem over \mathbb{K} . The equation $\sum_{i=1}^m \mu_i f_i = 1$ is called a *certificate of infeasibility* of the polynomial system.

EXAMPLE 3. Consider again the following set of polynomials from Example 1:

$$F := \{f_1 := x_1^2 - 1, f_2 := 2x_1x_2 + x_3, f_3 := x_1 + x_2, f_4 := x_1 + x_3\}.$$

We can abbreviate the infeasible polynomial system of equations $f_1(x) = 0, f_2(x) = 0, f_3(x) = 0, f_4(x) = 0$ as $F(x) = 0$. We can prove that the system $F(x) = 0$ is infeasible if we can find $\mu \in \mathbb{R}^4$ satisfying the following:

$$\begin{aligned} & \mu_1 f_1 + \mu_2 f_2 + \mu_3 f_3 + \mu_4 f_4 = 1 \\ \Leftrightarrow & \mu_1(x_1^2 - 1) + \mu_2(2x_1 x_2 + x_3) + \mu_3(x_1 + x_2) + \mu_4(x_1 + x_3) = 1 \\ \Leftrightarrow & \mu_1 x_1^2 + 2\mu_2 x_1 x_2 + (\mu_2 + \mu_4)x_3 + \mu_3 x_2 + (\mu_3 + \mu_4)x_1 - \mu_1 = 1. \end{aligned}$$

Then, equating coefficients on the left and right hand sides of the equation above gives the following linear system of equations:

$$\begin{aligned} -\mu_1 &= 1 \quad (1), & \mu_3 + \mu_4 &= 0 \quad (x_1), & \mu_3 &= 0 \quad (x_2), \\ \mu_3 + \mu_4 &= 0 \quad (x_3), & 2\mu_2 &= 0 \quad (x_1 x_2), & \mu_1 &= 0 \quad (x_1^2). \end{aligned}$$

We abbreviate this system as $\mu^T F = 1$. Even though $F(x) = 0$ is infeasible, the linear system $\mu^T F = 1$ is infeasible, and so, we have not found a certificate of infeasibility of $F(x) = 0$.

More formally, let $f_i = \sum_{\alpha \in \mathbb{N}^n} f_{i,\alpha} x^\alpha$ where only finitely many $f_{i,\alpha}$ are non-zero $i = 1, \dots, m$. Then, $\sum_{i=1}^m \mu_i f_i = 1$ if and only if $\sum_{i=1}^m \mu_i f_{i,0} = 1$ and $\sum_{i=1}^m \mu_i f_{i,\alpha} = 0$ for all $\alpha \in \mathbb{N}^n$ where $\alpha \neq 0$. Note that there is one linear equation per monomial appearing in F . We abbreviate this linear system as $\mu^T F = 1$ where we consider F as a matrix whose rows are the coefficient vectors of its polynomials and we consider the constant polynomial 1 as the vector of its coefficients (i.e., a unit vector). The columns of F are indexed by monomials with non-zero coefficients. We remark that in the special case where $F(x) = 0$ is a linear system of equations, then Fredholm's alternative says that $F(x) = 0$ is infeasible if and only if $\mu^T F = 1$ is feasible.

REMARK 2.1. Crucially for computation, when we solve the linear system $\mu^T F = 1$, we can do so over the smallest subfield of \mathbb{K} containing the coefficients of the polynomials in F , which is particularly useful if such a subfield is a finite field.

In general, even if $F(x) = 0$ is infeasible, $\mu^T F = 1$ may not be feasible as in the above example. In order to prove infeasibility, we must add polynomials from **ideal**(F) to F and try again to find a μ such that $\mu^T F = 1$. Hilbert's Nullstellensatz guarantees that, if $F(x) = 0$ is infeasible, there exists a finite set of polynomials from **ideal**(F) that we can add to F so that the linear system $\mu^T F = 1$ is feasible.

More precisely, it is enough to add polynomials of the form $x^\alpha f$ for x^α a monomial and some polynomial $f \in F$. Why is this? If $F(x) = 0$ is infeasible, then Hilbert's Nullstellensatz says $\sum_{i=1}^m \beta_i f_i = 1$ for some $\beta_1, \dots, \beta_m \in \mathbb{K}[x]$. Let $d = \max_i \{\deg(\beta_i)\}$. Then, if we add to F all polynomials of the form $x^\alpha f$ where $f \in F$ and $\deg(x^\alpha) \leq d$. Then, the \mathbb{K} -linear span of F , that is **span**(F), contains $\beta_i f_i$ for all i , and thus,

$1 \in \text{span}(F)$ or equivalently $\mu^T F' = 1$ is feasible (as a linear algebra problem) where F' denotes the larger polynomial system.

EXAMPLE 4. Consider again the polynomial system $F(x) = 0$ from Example 3. Here, $\mu^T F = 1$ is feasible, so we must thus add redundant polynomial equations to the system $F(x) = 0$. In particular, we add the following redundant polynomial equations: $x_2 f_1(x) = 0$, $x_1 f_2(x) = 0$, $x_1 f_3(x) = 0$, and $x_1 f_4(x) = 0$. Let $F' := \{f_1, f_2, f_3, f_4, x_2 f_1, x_1 f_2, x_1 f_3, x_1 f_4\}$.

Then, the system $\mu^T F' = 1$ is now as follows:

$$\begin{aligned} -\mu_1 &= 1 \quad (1), & \mu_3 + \mu_4 &= 0 \quad (x_1), & \mu_3 - \mu_5 &= 0 \quad (x_2), \\ \mu_2 + \mu_4 &= 0 \quad (x_3), & 2\mu_2 + \mu_7 &= 0 \quad (x_1 x_2), & \mu_1 + \mu_7 + \mu_8 &= 0 \quad (x_1^2), \\ \mu_6 + \mu_8 &= 0 \quad (x_1 x_3), & \mu_5 + 2\mu_6 &= 0 \quad (x_1^2 x_2). \end{aligned}$$

This system is feasible proving that $F(x) = 0$ is infeasible. The solution is $\mu = (-1, -\frac{2}{3}, -\frac{2}{3}, \frac{2}{3}, -\frac{2}{3}, -\frac{1}{3}, \frac{4}{3}, -\frac{1}{3})$, which gives the following certificate of infeasibility as given in Example 1:

$$-f_1 - \frac{2}{3}f_2 - \frac{2}{3}f_3 + \frac{2}{3}f_4 - \frac{2}{3}x_2 f_1 + \frac{1}{3}x_1 f_2 + \frac{4}{3}x_1 f_3 - \frac{1}{3}x_1 f_4 = 1.$$

Next, we present the dual approach to the one in this section.

2.2. Linear algebra relaxations. In optimization, it is quite common to “linearize” non-linear polynomial systems of equations by replacing all monomials in the system with new variables giving a system of linear constraints. Specifically, we can construct a linear algebra relaxation of the solutions of $F(x) = 0$ by replacing every monomial x^α in a polynomial equation in $F(x) = 0$ with a new variable λ_{x^α} thereby giving a system of linear equations in the new λ variables, one variable for each monomial appearing in F . Readers familiar with relaxation procedures such as Sherali-Adams and Lovász-Schrijver (see [27] and references therein) will see a lot of similarities, but here we deal only with *equality* constraints.

EXAMPLE 5. Consider the following feasible system in $\mathbb{C}[x_1, x_2, x_3]$:

$$f_1(x) = x_1^2 - 1 = 0, \quad f_2(x) = 2x_1 x_2 + x_3 = 0, \quad f_3(x) = x_1 + x_2 = 0.$$

This system has two solutions $(x_1, x_2, x_3) = (1, -1, 2)$ and $(x_1, x_2, x_3) = (-1, 1, 2)$. Let $F = \{f_1, f_2, f_3\}$. So, we abbreviate the above system as $F(x) = 0$. We can replace the monomials $1, x_1, x_2, x_3, x_1^2, x_1 x_2$ with the variables $\lambda_1, \lambda_{x_1}, \lambda_{x_2}, \lambda_{x_3}, \lambda_{x_1^2}, \lambda_{x_1 x_2}$ respectively. The system $F(x) = 0$ thus gives rise to the following set of linear equations:

$$\lambda_{x_1^2} - \lambda_1 = 0, \quad 2\lambda_{x_1 x_2} + \lambda_{x_3} = 0, \quad \lambda_{x_1} + \lambda_{x_2} = 0. \quad (2.1)$$

We abbreviate the above system as $F * \lambda = 0$.

Solutions of $F(x) = 0$ give solutions of $F * \lambda = 0$: If x is a solution of $F(x) = 0$ above, then setting $\lambda_1 = 1, \lambda_{x_1} = x_1, \lambda_{x_2} = x_2, \lambda_{x_3} = x_3, \lambda_{x_1^2} =$

$x_1^2, \lambda_{x_1 x_2} = x_1 x_2$ gives a solution of $F * \lambda = 0$. So, taking $x = (1, -1, 2)$, we set $\lambda_1 = 1$, $\lambda_{x_1} = 1$, $\lambda_{x_2} = -1$, $\lambda_{x_3} = 2$, $\lambda_{x_1^2} = 1$, and $\lambda_{x_1 x_2} = -1$. Then, we have $F * \lambda = 0$. Thus, the solutions of $F * \lambda = 0$ gives a vector space effectively containing all of the solutions of $F(x) = 0$. Hence, $F * \lambda = 0$ gives a linear relaxation of $F(x) = 0$.

There are solutions of $F * \lambda = 0$ that do not correspond to solutions of $F(x) = 0$ because the linear system $F * \lambda = 0$ does not take into account the non-linear constraints that $\lambda_1 = 1$, $\lambda_{x_1^2} = \lambda_{x_1}^2$ and $\lambda_{x_1 x_2} = \lambda_{x_1} \lambda_{x_2}$; For example, $\lambda_1 = 1$, $\lambda_{x_1} = 2$, $\lambda_{x_2} = -2$, $\lambda_{x_3} = -2$, $\lambda_{x_1^2} = 1$ and $\lambda_{x_1 x_2} = 1$ is a solution of $F * \lambda = 0$, but $x_1 = \lambda_{x_1} = 2$, $x_2 = \lambda_{x_2} = -2$, and $x_3 = \lambda_{x_3} = -2$ is not a solution of $F(x) = 0$.

We now formalize the above example construction of a linear system. We can consider the polynomial ring $\mathbb{K}[x]$ as an infinite dimensional vector space over \mathbb{K} where the set of all monomials x^α forms a vector space basis of $\mathbb{K}[x]$. In other words, a polynomial $f = \sum_{\alpha \in \mathbb{N}^n} f_\alpha x^\alpha$ can be represented as an infinite sequence $(f_\alpha)_{\alpha \in \mathbb{N}^n}$ where only finitely many f_α are non-zero. We define $\mathbb{K}[[x_1, \dots, x_n]] = \mathbb{K}[[x]]$ as the ring of formal power series in the variables x_1, \dots, x_n with coefficients in \mathbb{K} . So, the power series $\lambda = \sum_{\alpha \in \mathbb{N}^n} \lambda_\alpha x^\alpha$ can be represented as an infinite sequence $(\lambda_\alpha)_{\alpha \in \mathbb{N}^n}$. Note that we do not require that only finitely many λ_α are non-zero. We define the bilinear form $* : \mathbb{K}[x] \times \mathbb{K}[[x]] \rightarrow \mathbb{K}$ as follows: given $f = \sum_{\alpha \in \mathbb{N}^n} f_\alpha x^\alpha \in \mathbb{K}[x]$ and $\lambda = \sum_{\alpha \in \mathbb{N}^n} \lambda_\alpha x^\alpha \in \mathbb{K}[[x]]$, we define $f * \lambda = \sum_{\alpha \in \mathbb{N}^n} f_\alpha \lambda_\alpha$, which is always finite since only finitely many f_α are non-zero. Thus, we define a linear relaxation of $\{x \in \mathbb{K}^n : F(x) = 0\}$, written as $\{\lambda \in \mathbb{K}[[x]] : F * \lambda = 0\}$, as the set of linear equations $f * \lambda = 0$ for all $f \in F$. We denote the set of solutions of the linear system $F * \lambda = 0$ as $F^\circ := \{\lambda \in \mathbb{K}[[x]] : F * \lambda = 0\}$, called the *annihilator* of F , which is a vector subspace of $\mathbb{K}[[x]]$. See Appendix A for further details.

Note that, for any polynomial $f \in \mathbb{K}[x]$ and any point $v \in \mathbb{K}^n$, we have $f(v) = f * \lambda(v)$ where $\lambda(v) = (v^\alpha)_{\alpha \in \mathbb{N}^n}$. Thus, for any $v \in \mathbb{K}^n$, $F(v) = 0$ if and only if $F * \lambda(v) = 0$. So, the system $F * \lambda = 0$ can be considered as a linear relaxation of the system $F(x) = 0$. As mentioned in the above example, there are solutions of $F * \lambda = 0$ that do not correspond to solutions of $F(x) = 0$ because the linear system $F * \lambda = 0$ does not take into account the relationships between the λ variables. Specifically, if λ corresponded to a solution of $F(x) = 0$, then we must have $\lambda_{x^\alpha} = \lambda_{x^\beta} \lambda_{x^\gamma}$ for all monomials $x^\alpha, x^\beta, x^\gamma$ where $x^\alpha = x^\beta x^\gamma$. If we added these non-linear constraints to the linear constraints $F * \lambda = 0$, then we would essentially have the original polynomial system $F(x) = 0$.

The system $F * \lambda = 0$ is always feasible, but the constraint $\lambda_1 = 1$ also holds for any λ that corresponds to a solution x of $F(x) = 0$. Thus, if the inhomogeneous linear system $\{F * \lambda = 0, \lambda_1 = 1\}$ is infeasible, then so is the system of polynomials $F(x) = 0$.

REMARK 2.2. Crucially for computation again, when we solve the linear system $\{F * \lambda = 0, \lambda_1 = 1\}$, we can do so over the smallest subfield of \mathbb{K} containing the coefficients of the polynomials in F .

REMARK 2.3. Importantly, the linear system $\{F * \lambda = 0, \lambda_1 = 1\}$ is dual to the linear system $\mu^T F = 1$ from the previous section by Fredholm's alternative meaning that $\{F * \lambda = 0, \lambda_1 = 1\}$ is infeasible if and only if $\mu^T F = 1$ is feasible.

There is a fundamental observation we wish to make here: *adding redundant polynomial equations can lead to a tighter relaxation.*

EXAMPLE 6. (*Cont.*) Add $x_1 f_3(x) = x_1^2 + x_1 x_2 = 0$ to the system $F(x) = 0$ giving the system $F'(x) = 0$ where $F' := \{f_1, f_2, f_3, x_1 f_3\}$. The system $F'(x) = 0$ has the same solutions as $F(x) = 0$. The polynomial equation $x_1 f_3(x) = 0$ gives rise to a new linear equation $\lambda_{x_1^2} + \lambda_{x_1 x_2} = 0$ giving the following linear system $F' * \lambda = 0$:

$$\lambda_{x_1^2} - \lambda_1 = 0, 2\lambda_{x_1 x_2} + \lambda_{x_3} = 0, \lambda_{x_1} + \lambda_{x_2} = 0, \lambda_{x_1^2} + \lambda_{x_1 x_2} = 0. \quad (2.2)$$

The dimension of the solution space of the original system $F * \lambda = 0$ is three if we ignore all λ variables that do not appear in the linear system, or in other words, if we project the solution space onto the λ variables appearing in the system. However, the dimension of the projected solution space of $F' * \lambda = 0$ is two; so, $F' * \lambda = 0$ is a tighter relaxation of $F(x) = 0$.

Extending this idea, consider the ideal $I = \text{ideal}(F)$, which is the set of all redundant polynomials given as a polynomial combination of polynomials in F , then I° becomes a finite dimensional vector space where $\dim(I^\circ)$ is precisely the number of solutions of $F(x) = 0$ over \mathbb{K} , including multiplicities, assuming that there are finitely many solutions. Note that by linear algebra, I° is isomorphic to the vector space quotient $\mathbb{K}[x]/I$ (see e.g., [50]). Furthermore, if I is radical, then $\dim(I^\circ) = \dim(\mathbb{K}[x]/I)$ is precisely the number of solutions of $F(x) = 0$. So, there is a direct relationship between the number of solutions of a polynomial system and the dimension of the solution space of its linear relaxation (see e.g., [6] for a proof).

THEOREM 2.1. Let $I \subseteq \mathbb{K}[x]$ be a zero-dimensional ideal. Then, $\dim(I^\circ)$ is finite and $\dim(I^\circ)$ is the number of solutions of polynomial system $I(x) = 0$ over \mathbb{K} including multiplicities, so $|\mathcal{V}_\mathbb{K}(I)| \leq \dim(I^\circ)$ with equality when I is radical.

So, if we can compute $\dim(I^\circ)$, then we can determine the feasibility of $I(x) = 0$ over \mathbb{K} . Unfortunately, we cannot compute $\dim(I^\circ)$ directly. Instead, under some conditions (see Theorem 2.2), we can compute $\dim(I^\circ)$ by computing the dimension of F° when projected onto the λ_{x^α} variables where $\deg(x^\alpha) \leq \deg(F)$.

2.3. Nullstellensatz Linear Algebra Algorithm (NulLA). We now present an algorithm for determining whether a polynomial system of equations is infeasible using linear relaxations. Let $F \subseteq \mathbb{K}[x]$ and again let

$F(x) = 0$ be the polynomial system $f(x) = 0$ for all $f \in F$. We wish to determine whether $F(x) = 0$ has a solution over \mathbb{K} .

The idea behind NulLA [8] is straightforward: we check whether the linear system $\{F * \lambda = 0, \lambda_1 = 1\}$ is infeasible or equivalently whether $\mu^T F = 1$ is feasible (i.e., $1 \in \text{span}(F)$) using linear algebra over \mathbb{K} and if not then we add polynomials from $\text{ideal}(F)$ to F and try again. We add polynomials in the following systematic way: for each polynomial $f \in F$ and for each variable x_i , we add $x_i f$ to F . So, the NulLA algorithm is as follows: if $\{F * \lambda = 0, \lambda_1 = 1\}$ is infeasible, then $F(x) = 0$ is infeasible and stop, otherwise for every variable x_i and every $f \in F$ add $x_i f$ to F and repeat.

In the following, we assume without loss of generality that F is closed under \mathbb{K} -linear combinations, that is $F = \text{span}(F)$, and thus, F is a vector space over \mathbb{K} . Note that taking the closure of F under \mathbb{K} -linear combinations does not change the set of solutions of $F(x) = 0$ and does not change the set of solutions of $F * \lambda = 0$. In practice, we must choose a vector space basis of F for computation, but the point we wish to make is that the choice of basis is irrelevant. Moreover, we find that it is more natural to work with vector spaces and that it leads to a more concise exposition. Recall from above that $\{F * \lambda = 0, \lambda_1 = 1\}$ is infeasible if and only if $1 \in \text{span}(F)$, which when F is a vector space, simplifies to $1 \in F$ since $\text{span}(F) = F$.

For a vector space $F \subset \mathbb{K}[x]$, we define $F^+ := F + \sum_{i=1}^n x_i F$ where $x_i F := \{x_i f : f \in F\}$. Note that F^+ is also a vector subspace of $\mathbb{K}[x]$. Then, F^+ is precisely the linear span of F and $x_i F$ for all $i = 1, \dots, n$. So, the NulLA algorithm for vector spaces is as follows (see Algorithm 1): if $1 \in F$, then $F(x) = 0$ is infeasible and stop, otherwise set $F \leftarrow F^+$ and repeat. There is an upper bound on the number of times we need to repeat the above step given by the *Nullstellensatz bound* of the system $F(x) = 0$ (see [22]): if $F(x) = 0$ has a Nullstellensatz bound D , then if $F(x) = 0$ is infeasible, there must exist a Nullstellensatz certificate of infeasibility $\sum_i \beta_i f_i = 1$ where $\deg(\beta_i) \leq D$, that is, the degree of the certificate is at most $\deg(F) + D$. After d iterations of NulLA, the set F contains all linear combinations of polynomials of the form $x^\alpha f$ where $|\alpha| \leq d$ and where f was one of the initial polynomials in F , and so, if the system is infeasible, then NulLA will find a certificate of infeasibility in at most the Nullstellensatz bound number of iterations.

While theoretically the Nullstellensatz bound limits the number of iterations, this bound is in general too large to be practically useful (see [8]). Hence, in practice, NulLA is most useful for proving infeasibility (see Section 2.4).

Next, we discuss improving NulLA by adding redundant polynomials to F in such a way so that $\deg(F)$ does not grow unnecessarily. We call this improved algorithm the Fixed-Point Nullstellensatz Linear Algebra (FPNulLA) algorithm. Some variations of FPNulLA appeared, e.g., in

Algorithm 1 NulLA Algorithm [8]

Input: A finite dimensional vector space $F \subseteq \mathbb{K}[x]$ and a Nullstellensatz bound D .

Output: FEASIBLE, if $F(x) = 0$ is feasible over \mathbb{K} , else INFEASIBLE.

- 1: **for** $k = 0, 1, 2, \dots, D$ **do**
 - 2: If $1 \in F$, then **return** INFEASIBLE.
 - 3: $F \leftarrow F^+$.
 - 4: **end for**
 - 5: **Return** FEASIBLE.
-

[37, 44, 25]. The basic idea behind the FPNuLLA algorithm is that, if $1 \notin F$, then instead of replacing F with F^+ and thereby increasing $\deg(F)$, we check to see whether there are any new polynomials in F^+ with degree at most $\deg(F)$ that were not in F and add them to F , and then check again whether $1 \notin F$. More formally, if $1 \notin F$, then we replace F with $F^+ \cap \mathbb{K}[x]_d$ where $\mathbb{K}[x]_d$ is the set of all polynomials with degree at most $d = \deg(F)$. We keep replacing F with $F^+ \cap \mathbb{K}[x]_d$ until either $1 \in F$ or we reach a *fixed point*, $F = F^+ \cap \mathbb{K}[x]_d$. This process must terminate.

Note that if we find that $1 \in F$ at some stage of FPNuLLA this implies that there exists an infeasibility certificate of the form $1 = \sum_{i=1}^s \beta_i f_i$ where $\beta_1, \dots, \beta_s \in \mathbb{K}[x]$ and the polynomials $f_1, \dots, f_s \in \mathbb{K}[x]$ are a vector space basis of the original set F .

Moreover, we can also improve NuLLA by proving that the system $F(x) = 0$ is feasible well before reaching the Nullstellensatz bound as follows. When $1 \notin F$ and $F = F^+ \cap \mathbb{K}[x]_d$, then we could set $F \leftarrow F^+$ and $d \leftarrow d + 1$ and repeat the above process. However, when we reach the fixed point $F = F^+ \cap \mathbb{K}[x]_d$, we can use the following theorem to determine if the system is feasible and if so how many solutions it has. First, we introduce some notation. Let $\pi_d : \mathbb{K}[[x]] \rightarrow \mathbb{K}[[x]]_d$ be the truncation or projection of a power series onto a polynomial of degree at most d with coefficients in \mathbb{K} . Below, we abbreviate $\dim(\pi_d(F^\circ))$ as $\dim_d(F^\circ)$ and similarly $\dim(\pi_{d-1}(F^\circ))$ as $\dim_{d-1}(F^\circ)$.

THEOREM 2.2. *Let $F \subset \mathbb{K}[x]$ be a finite dimensional vector space and let $d = \deg(F)$. If $F = F^+ \cap \mathbb{K}[x]_d$ and $\dim_d(F^\circ) = \dim_{d-1}(F^\circ)$, then $\dim(I^\circ) = \dim_d(F^\circ)$ where $I = \mathbf{ideal}(F)$.*

See the Appendix for a proof of Theorem 2.2 or see original proof in [37]. There are many equivalent forms of the above theorem that appear in the literature (see e.g., [37, 44, 25]).

Recall from Theorem 2.1, that there are $\dim(I^\circ)$ solutions of $F(x) = 0$ over \mathbb{K} including multiplicities where $I = \mathbf{ideal}(F)$ and exactly $\dim(I^\circ)$ solutions when I is radical. Checking the fixed point condition in FPNuLLA whether $F \neq F^+ \cap \mathbb{K}[x]_d$ is equivalent to checking whether $\dim(F) \neq \dim(F^+ \cap \mathbb{K}[x]_d)$. Furthermore, to check the condition that $\dim_d(F^\circ) = \dim_{d-1}(F^\circ)$, we need to compute $\dim(F^+ \cap \mathbb{K}[x]_d)$ and $\dim(F \cap \mathbb{K}[x]_{d-1})$

since $\dim(\mathbb{K}[x]_d/F) = \dim_d(F^\circ)$ and also $\dim(\mathbb{K}[x]_{d-1}/(F \cap \mathbb{K}[x]_{d-1})) = \dim_{d-1}(F^\circ)$ (see Lemma A.1). So, in order to check the condition in FP-NuLA, we need to compute $\dim(F)$, $\dim(F^+ \cap \mathbb{K}[x]_d)$ and $\dim(F \cap \mathbb{K}[x]_{d-1})$, which amounts to matrix rank calculations over the field of coefficients of a given basis of F .

We can now present the FPNuLA algorithm. See the Appendix or [37, 7] for details. The FPNuLA algorithm always terminates for zero-dimensional polynomial systems, which in particular includes combinatorial systems (see Lemma A.2).

Algorithm 2 FPNuLA Algorithm

Input: A vector space $F \subset \mathbb{K}[x]$.

Output: The number of solutions of $F(x) = 0$ over \mathbb{K} up to multiplicities.

- 1: Let $d \leftarrow \deg(F)$.
 - 2: **loop**
 - 3: **if** $1 \in F$ **then** Return 0 (infeasible).
 - 4: **while** $F \neq F^+ \cap \mathbb{K}[x]_d$ **do**
 - 5: Set $F \leftarrow F^+ \cap \mathbb{K}[x]_d$.
 - 6: **if** $1 \in F$ **then** Return 0 (infeasible).
 - 7: **end while**
 - 8: **if** $\dim_d(F^\circ) = \dim_{d-1}(F^\circ)$ **then** Return $\dim_d(F^\circ)$ (feasible).
 - 9: $F \leftarrow F^+$.
 - 10: $d \leftarrow d + 1$.
 - 11: **end loop**
-

EXAMPLE 7. Consider again the system below with polynomials in $\mathbb{K}[x, y]$ with $\mathbb{K} = \overline{\mathbb{F}}_2$. This system has two solutions.

$$1 + x + x^2 = 0, \quad 1 + y + y^2 = 0, \quad x^2 + xy + y^2 = 0.$$

Let $F := \text{span}(\{1 + x + x^2, 1 + y + y^2, x^2 + xy + y^2\})$. Then, $1 \notin F$ and $\deg(F) = 2$. Now,

$$\begin{aligned} F^+ &= F + xF + yF \\ &= F + \text{span}(\{x + x^2 + x^3, x + xy + xy^2, x^3 + x^2y + xy^2\}) \\ &\quad + \text{span}(\{y + xy + x^2y, y + y^2 + y^3, x^2y + xy^2 + y^3\}). \end{aligned}$$

Then, $F^+ \cap \mathbb{K}[x]_2 = \text{span}(\{1 + x + x^2, 1 + y + y^2, x^2 + xy + y^2, 1 + x + y\})$. So, $F \neq F^+ \cap \mathbb{K}[x]_2$. Next, let $F := F^+ \cap \mathbb{K}[x]_2$. One can check that now $F = F^+ \cap \mathbb{K}[x]_2$. Moreover,

$$\dim_2(F^\circ) = \dim(\mathbb{K}[x]_2/F) = \dim(\mathbb{K}[x]_2) - \dim(F) = 2$$

and

$$\dim_1(F^\circ) = \dim(\mathbb{K}[x]_1/(F \cap \mathbb{K}[x]_1)) = \dim(\mathbb{K}[x]_1) - \dim(F \cap \mathbb{K}[x]_1) = 2.$$

Therefore, $\dim_2(F^\circ) = \dim_1(F^\circ)$ proving that $F(x) = 0$ is feasible with at most 2 solutions.

We refer to the number of iterations (the *for loop*) that NullA takes to solve a given system of equations as the *NullA rank* of the system. Note that if an infeasible system $F(x) = 0$ has a NullA rank of r , then it has a Nullstellensatz certificate of infeasibility of degree $r + \deg(F)$. Similarly to the NullA rank, we refer to the number of outer iterations (the *outer loop*) that FPNulLA takes to the system as the *FPNullA rank* of the system. We can consider the NullA rank and the FPNulLA rank as measures of the “hardness” of proving infeasibility of the system. In section 2.4, we present experimental evidence that the NullA rank and even more so the FPNulLA are “good” measures of the “hardness” of proving infeasibility of a system (see also [3] for theoretical evidence for FPNulLA).

For a given class of polynomial system of equations, it is interesting to understand the growth of the NullA rank or FPNulLA rank because of the implications for the complexity of solving the given class of problems. Furthermore, for some fixed rank, it is also interesting to characterize which systems can be solved at that rank since this class of systems are polynomial time solvable by Lemma 2.1 below (see proof in Appendix and a proof for NullA in [35]). For example, in Section 2.5, we characterize systems encoding 3-colorability with NullA rank one.

LEMMA 2.1. *Let $L \in \mathbb{N}$ be fixed. Let $F = \text{span}(\{f_1, f_2, \dots, f_m\}) \subseteq \mathbb{K}[x]$ be a finite dimensional vector space of $\mathbb{K}[x]$. Polynomials are assumed to be encoded as vectors of coefficients indexed by all monomials of degree at most $\deg(F)$.*

1. *The first L iterations (the for loop) of the NullA algorithm can be computed in polynomial time in n and the input size of the defining basis of F .*
2. *When \mathbb{K} is a finite field, the first L iterations (the outer loop) of the FPNulLA algorithm can be computed in polynomial time in n , $\log_2(|\mathbb{K}|)$ and the input size of the defining basis of F .*

2.4. Experimental results. In this section, we summarize experimental results for graph 3-coloring from [7], which illustrate the practical performance of the NullA and FPNulLA algorithms. For further and more detailed results, see [8, 35, 7]. Experimentally, for graph 3-coloring, NullA and FPNulLA are well-suited to proving infeasibility, that is, that no 3-coloring exists. The system polynomials we use to encode 3-colorability has coefficients on \mathbb{F}_2 (see Proposition 1.1) and thus the linear algebra operations are very fast. However, even though in theory NullA and FPNulLA can determine feasibility, for the experiments described below NullA and FPNulLA are only suitable for proving infeasibility.

Here, we are interested in the percentage of randomly generated graphs whose polynomial system encoding has a NullA rank of one, a NullA rank of two or a FPNulLA rank of one. The $G(n, p)$ model [13] is used

for generating random graphs where n is the number of vertices and p is the probability that an edge is included between any two vertices. Also, without loss of generality, for a slightly smaller polynomial encoding, the color of one of the vertices of each randomly generated graph was fixed.

The experimental results are presented in [Figure 1](#) (taken from [7]), which plots the percentage of 1000 random graphs in $G(100, p)$ that were proven infeasible with a NullA rank of one, with a NullA rank of two, with a FPNuLA rank of one, or with an exact method versus the p value. The exact method used was to model graph 3-coloring as a Boolean satisfiability problem [12] and then use the program `zchaff` [52] to solve the satisfiability problem.

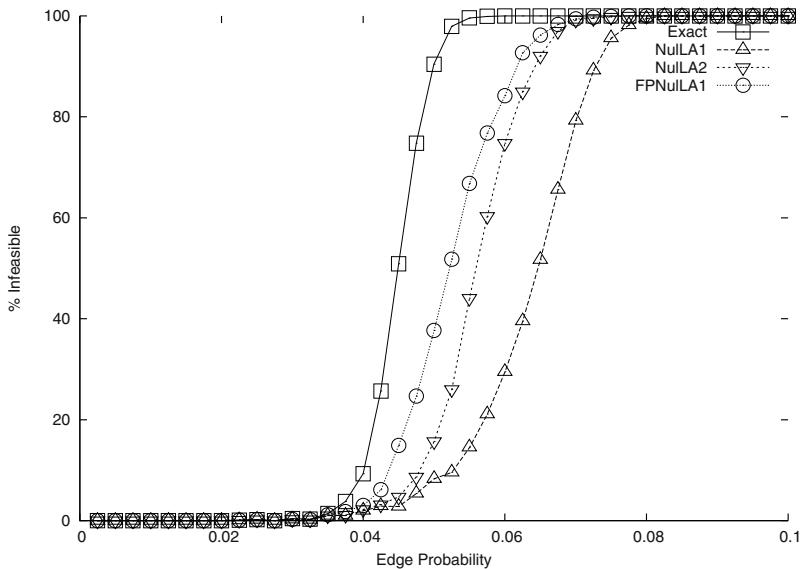


FIG. 1. Non-3-colorable graphs with NullA rank 1 and 2 and FPNuLA rank 1.

It is well-known that there is a distinct phase transition from feasibility to infeasibility for graph 3-coloring, and it is at this phase transition that graphs exist for which it is difficult on average to prove infeasibility or feasibility (see [19]). Observe that the infeasibility curve for NullA resembles that of the exact infeasibility curve and that the infeasibility curve for FPNuLA also resembles the infeasibility curve and clearly dominates the infeasibility curve for NullA. These results suggest that the NullA rank or FPNuLA rank are a reasonable measure of the hardness of proving infeasibility since those graphs that require a high rank are located near the phase transition.

Lastly, we comment on the runnings times of NullA and FPNuLA and the exact approach using `zchaff` for the experiments on random graphs in

$G(100, p)$ above. The NulLA rank one and FPNulLA rank one approaches ran on average in less than a second for all p values. However, the exact approach using `zchaff` ran in split second times for all p values, but preliminary computational experiments indicate that the gap in running times between the exact approach and the FPNulLA rank one approach closes for larger graphs. The NulLA rank two approach ran on average in less than a second for $p \leq 0.04$ and $p \geq 0.08$, but the average running times peaked at about 24 seconds at $p = 0.65$. Interestingly, for each approach, the average running time peaked at the transition from feasible to infeasible at the p value where about half of the graphs were proven infeasible by the approach.

In order to better understand the practical implications of the NullA and FPNulLA approaches, there needs to be more detailed computational studies performed to compare this approach with the exact method using satisfiability and other exact approaches such as traditional integer programming techniques. See [8] for some additional experimental data.

2.5. Application: The structure of non-3-colorable graphs. In this section, we state a combinatorial characterization of those graphs whose combinatorial system of equations encoding 3-colorability has a NullA rank of one thus giving a class of polynomial solvable graphs by Lemma 2.1, and also, we recall bounds for the NullA rank (see [35]):

THEOREM 2.3. *The NullA rank for a polynomial encoding over \mathbb{F}_2 of the 3-colorability of a graph with n vertices with no 3-coloring is at least one and at most $2n$. Moreover, in the case of a non-3-colorable graph containing an odd-wheel (e.g. a 4-clique) as a subgraph, the NullA rank is exactly one.*

Now we look at those non-3-colorable graphs that have a NullA rank of one. Let A denote the set of all possible directed edges or arcs in the graph G . We are interested in two types of substructures of the graph G : oriented partial-3-cycles and oriented chordless 4-cycles (see Figure 2). An **oriented partial-3-cycle** is a set of two arcs of a 3-cycle, that is, a set $\{(i, j), (j, k)\}$ also denoted (i, j, k) where $(i, j), (j, k), (k, i) \in A$. An **oriented chordless 4-cycle** is a set of four arcs $\{(i, j), (j, l), (l, k), (k, i)\}$ also denoted (i, j, k, l) where $(i, j), (j, l), (l, k), (k, i) \in A$ and $(j, k), (i, l) \notin A$.

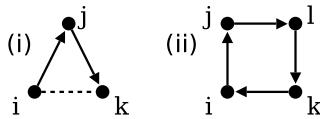


FIG. 2. (i) oriented partial 3-cycle and (ii) an oriented chordless 4-cycle.

Now, we can state a sufficient condition for non-3-colorability [7]. This sufficient condition is satisfied if and only if the combinatorial system encoding 3-coloring has a NullA rank of one, which is proved in [7].

THEOREM 2.4. *The graph G is not 3-colorable if there exists a set C of oriented partial 3-cycles and oriented chordless 4-cycles such that*

1. $|C_{(i,j)}| + |C_{(j,i)}| \equiv 0 \pmod{2}$ for all $(i, j) \in E$ and
2. $\sum_{(i,j) \in A, i < j} |C_{(i,j)}| \equiv 1 \pmod{2}$

where $|C_{(i,j)}|$ denotes the number of cycles in C (either 3-cycles or 4-cycles) in which the arc $(i, j) \in A$ appears.

Condition 1 in Theorem 2.4 means that every undirected edge of G is covered by an even number of directed edges from cycles in C (ignoring orientation). Condition 2 in Theorem 2.4 means that, given any orientation of G , the total number of times the arcs in that orientation appear in the cycles of C is odd. The particular orientation we use in Theorem 2.4 is the orientation given by the set of arcs $\{(i, j) \in A : i < j\}$, but the particular orientation we use for Condition 2 is irrelevant (see [7]).

Using Theorem 2.4, proving that graphs containing odd wheels (e.g., 4-cliques) are not 3-colorable (see Theorem 2.3) is straight-forward ([7]):

EXAMPLE 8. *Assume a graph G contains an odd wheel with vertices labelled as in Figure 3 below. Consider the following set of oriented partial 3-cycles: $C := \{(i, 1, i+1) : 2 \leq i \leq n-1\} \cup \{(n, 1, 2)\}$. The oriented partial 3-cycles of C are shown in Figure 3.*

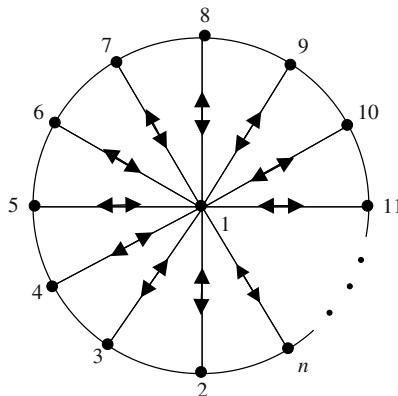


FIG. 3. Odd wheel.

The set C satisfies Condition 1 of Theorem 2.4 since each edge is covered by exactly zero or two cycles in C . Also, C satisfies Condition 2 of Theorem 2.4 since each arc $(1, i) \in \text{Arcs}(G)$ is covered exactly once by a cycle in C and there are an odd number of arcs $(1, i) \in \text{Arcs}(G)$. Thus, G is non-3-colorable by Theorem 2.4.

The Grötzsch graph is a non-trivial example of a non-3-colorable graph with a degree one Nullstellensatz certificate ([7]):

EXAMPLE 9. *Consider the Grötzsch graph (Mycielski 4) in Figure 4, which has no 3-coloring. It contains no 3-cycles. Now, consider the*

following set of oriented chordless 4-cycles, which we show gives a certificate of non-3-colorability by Theorem 2.4.

$$C := \{(1, 2, 3, 7), (2, 3, 4, 8), (3, 4, 5, 9), (4, 5, 1, 10), (1, 10, 11, 7), (2, 6, 11, 8), (3, 7, 11, 9), (4, 8, 11, 10), (5, 9, 11, 6)\}.$$

Figure 4 illustrates the edge directions for the 4-cycles of C . Each undirected edge of the graph is contained in exactly two 4-cycles, so C satisfies Condition 1 of Theorem 2.4. Now,

$$|C_{(6,11)}| = |C_{(7,11)}| = |C_{(8,11)}| = |C_{(9,11)}| = |C_{(10,11)}| = 1,$$

and $|C_{(i,j)}| \equiv 0 \pmod{2}$ for all other arcs $(i,j) \in A$ where $i < j$. Thus,

$$\sum_{(i,j) \in A, i < j} |C_{(i,j)}| \equiv 1 \pmod{2},$$

so Condition 2 is satisfied, and therefore, the graph has no 3-coloring.

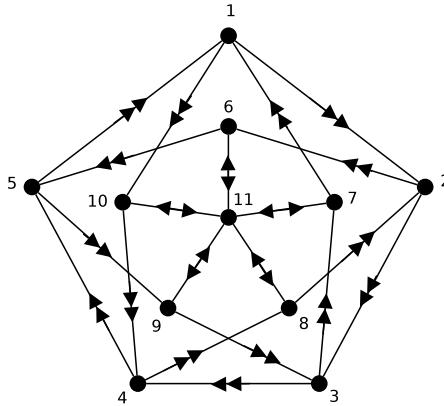


FIG. 4. Grötzsch graph

There are no known combinatorial characterizations concerning higher NullA ranks.

3. Adding polynomial inequalities. Up until this point we have worked over arbitrary fields (with special attention to finite fields due to their fast and exact computation), where the only allowable constraints were equations. Now we turn our attention to the real case (i.e. $\mathbb{K} = \mathbb{R}$), where we have the additional possibility of specifying *inequalities* (more generally, one can work over *ordered* or *formally real* fields). In this case, following the terminology of real algebraic geometry, we call the solution set of a system of polynomial equations and inequalities a *basic semialgebraic set*. Note that convex polyhedra correspond to the particular case where

all the constraint polynomials have degree one. As we have seen earlier in the Positivstellensatz (Theorem 1.4 above), the emptiness of a basic semialgebraic set can be certified through an algebraic identity involving sum of squares of polynomials.

The connection between sum of squares decompositions of polynomials and convex optimization can be traced back to the work of N. Z. Shor [48]. His work went relatively unnoticed for several years, until several authors, including Lasserre, Nesterov, and Parrilo, observed, around the year 2000, that the existence of sum of squares decompositions and the search for infeasibility certificates for a semialgebraic set can be addressed via a sequence of semidefinite programs relaxations [23, 40, 41, 39]. The first part of this section will be a short description of the connections between sums of squares and semidefinite programming, and how the Positivstellensatz allows, in a analogous way to what was presented in Section 2 for the Nullstellensatz, for a systematic way to formulate these semidefinite relaxations.

A very central preoccupation of combinatorial optimizers has been the understanding of the facets that describe the integer hull (normally binary) of a combinatorial problem. As we will see later on, one can recover quite a bit of information about the integer hull of combinatorial problems from a sequence combinatorially controlled SDPs. This kind of approach was pioneered in the lift-and-project method of Balas, Ceria and Cornuéjols [1], the matrix-cut method of Lovász and Schrijver [34] and the linearization technique of Sherali-Adams [47]. Here we try to present more recent developments (see [30] and references therein for a very extensive survey).

3.1. Sums of squares, SDP, and feasibility of semialgebraic sets. Recall that a multivariate polynomial $p(x)$ is a *sum of squares* (SOS for short) if it can be written as a sum of squares of other polynomials, that is, $p(x) = \sum_i q_i^2(x)$, $q_i(x) \in \mathbb{R}[x]$. The condition that a polynomial is a sum of squares is a quite natural sufficient test for polynomial non-negativity. Thus instead of asking whether even degree polynomials are non-negative we ask the easier question whether they are sums of squares. More importantly, as we shall see, the existence of a sum of squares decomposition can be decided via semidefinite programming.

THEOREM 3.1. *A polynomial $p(x)$ is SOS if and only if $p(x) = z^T Q z$, where z is a vector of monomials in the x_i variables, and Q is a symmetric positive semidefinite matrix.*

By the theorem above, every SOS polynomial can be written as a quadratic form in a set of monomials, with the corresponding matrix being positive semidefinite. The vector of monomials z in general depends on the degree and sparsity pattern of $p(x)$. If $p(x)$ has n variables and total degree $2d$, then z can always be chosen as a subset of the set of monomials of degree less than or equal to d , which has cardinality $\binom{n+d}{d}$.

EXAMPLE 10. *The polynomial $p(x_1, x_2) = x_1^2 - x_1x_2^2 + x_2^4 + 1$ is SOS. Among infinitely many others, $p(x_1, x_2)$ has the following decompositions:*

$$\begin{aligned} p(x_1, x_2) &= \frac{3}{4}(x_1 - x_2^2)^2 + \frac{1}{4}(x_1 + x_2^2)^2 + 1 \\ &= \frac{1}{9}(3 - x_2^2)^2 + \frac{2}{3}x_2^2 + \frac{1}{288}(9x_1 - 16x_2^2)^2 + \frac{23}{32}x_1^2. \end{aligned}$$

The polynomial $p(x_1, x_2)$ has the following representation:

$$p(x_1, x_2) = \frac{1}{6} \begin{bmatrix} 1 \\ x_2 \\ x_2^2 \\ x_1 \end{bmatrix}^T \begin{bmatrix} 6 & 0 & -2 & 0 \\ 0 & 4 & 0 & 0 \\ -2 & 0 & 6 & -3 \\ 0 & 0 & -3 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ x_2 \\ x_2^2 \\ x_1 \end{bmatrix}$$

where the matrix in the expression above is positive semidefinite.

In the representation $f(x) = z^T Q z$, for the right- and left-hand sides to be identical, all the coefficients of the corresponding polynomials should be equal. Since Q is simultaneously constrained by linear equations and a positive semidefiniteness condition, the problem can be easily seen to be directly equivalent to a semidefinite programming feasibility problem in the standard primal form.

Now we describe an algorithm (originally presented in [40, 41]) and illustrate it with an example, on how we can use SDPs to decide the feasibility of a system of polynomial inequalities. Exactly as we did for the Nullstellensatz case, we can look for the existence of a Positivstellensatz certificate of bounded degree D (see Theorem 1.4). Once we assume that the degree D is fixed we can apply Theorem 3.1 and obtain a reformulation as a semidefinite programming problem. We formalize this description in the following algorithm:

Algorithm 3 Bounded degree Positivstellensatz [40, 41]

Input: A polynomial system $\{f_i(x) = 0, g_i(x) \geq 0\}$ and a Positivstellensatz bound D .

Output: FEASIBLE, if $\{f_i(x) = 0, g_i(x) \geq 0\}$ is feasible over \mathbb{R} , else INFEASIBLE.

for $d = 0, 1, 2, \dots, D$ **do**

If there exist $\beta_i, s_\alpha \in \mathbb{R}[x]$ such that $-1 = \sum_i \beta_i f_i + \sum_{\alpha \in \{0,1\}^n} s_\alpha g^\alpha$,
with s_α SOS, $\deg(\beta_i f_i) \leq d$, $\deg(s_\alpha g^\alpha) \leq d$ then **return** INFEASIBLE.
 $d \leftarrow d + 1$.

end for

Return FEASIBLE.

Notice that the membership test in the main loop of the algorithm is, by the results described at the beginning of this section, equivalent to a finite-sized semidefinite program. Similarly to the Nullstellensatz case,

the number of iterations (i.e., the degree of the certificates) serves as a quantitative measure of the hardness in proving infeasibility of the system. As we will describe in more detail in Section 3.4, in several situations one can give further refined characterization on these degrees.

EXAMPLE 11. Consider the polynomial system $\{f = 0, g \geq 0\}$ from Example 2, where $f := x_2 + x_1^2 + 2 = 0$ and $g := x_1 - x_2^2 + 3 \geq 0$. At the d -th iteration of Algorithm 3 applied to the polynomial problem $\{f = 0, g \geq 0\}$, one asks whether there exist polynomials $\beta, s_1, s_2 \in \mathbb{K}[x]$ such that $\beta f + s_1 + s_2 \cdot g = -1$ where s_1, s_2 are SOS and $\deg(s_1), \deg(s_2 \cdot g), \deg(\beta \cdot f) \leq d$. For each fixed positive integer d this can be tested by a (possibly large) semidefinite program.

Solving this for $d = 2$, we have $\deg(s_1) \leq 2$, $\deg(s_2) = 0$ and $\deg(\beta) = 0$, so s_2 and β are constants and

$$\begin{aligned} s_1 = z^T Q z &= \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{12} & Q_{22} & Q_{23} \\ Q_{13} & Q_{23} & Q_{33} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \\ &= Q_{11} + 2Q_{12}x_1 + 2Q_{13}x_2 + Q_{22}x_1^2 + 2Q_{23}x_1x_2 + Q_{33}x_2^2 \end{aligned}$$

where $z = (1, x_1, x_2)^T$ and $Q \in \mathbb{R}^{3 \times 3}$ is a symmetric positive semidefinite matrix. Thus, the certificate for $D = 2$ is $\beta f + z^T Q z + s_2 \cdot g = -1$ where $Q \succeq 0$ and $s_2 \geq 0$. If we expand the left hand side and equate coefficients on both sides of the equation, we arrive at the following SDP:

$$\begin{aligned} 2\beta + Q_{11} + 3s_2 &= -1 \quad (1), & 2Q_{12} + s_2 &= 0 \quad (x_1), \\ \beta + 2Q_{13} &= 0 \quad (x_2), & \beta + Q_{22} &= 0 \quad (x_1^2), \\ 2Q_{23} &= 0 \quad (x_1x_2), & Q_{33} - s_2 &= 0 \quad (x_2^2) \end{aligned}$$

where $Q \succeq 0$ and $s_2 \geq 0$. This SDP has a solution as follows:

$$Q = \begin{bmatrix} 5 & -1 & 3 \\ -1 & 6 & 0 \\ 3 & 0 & 2 \end{bmatrix}, \quad s_2 = 2 \text{ and } \beta = -6.$$

The resulting identity, which is the same as the one given in Example 2, proves the inconsistency of the system.

As outlined in the preceding paragraphs, there is a direct connection going from general polynomial optimization problems to SDP, via the Positivstellensatz infeasibility certificates. Even though we have discussed only feasibility problems here, there are obvious straightforward connections with optimization. For instance, by considering the emptiness of the sublevel sets of the objective function, or using representation theorems for positive polynomials, sequences of converging bounds indexed by certificate degree can be directly constructed; see e.g. [40, 23, 42]. These schemes have been implemented in software packages such as SOSTOOLS [43], GloptiPoly [17], and YALMIP [31].

3.2. Semidefinite programming relaxations. In the last section, we have described the search for Positivstellensatz infeasibility certificates formulated as a semidefinite programming problem. We now describe an alternative interpretation, obtained by dualizing the corresponding semidefinite programs. This is the exact analogue of the construction presented in Section 2.2, and is closely related to the approach via truncated moment sequences developed by Lasserre [23].

Recall that in the approach in Section 2.2, the linear relaxations were constructed by replacing every monomial x^α by a new variable λ_{x^α} . Furthermore, new redundant equations were obtained by multiplying an existing constraint $f(x) = 0$ by terms of the form x_i , yielding $x_i f(x) = 0$ (essentially, generating the ideal of valid equations). In the inequality case, and as suggested by the Positivstellensatz, new inequality constraints will be generated by both squarefree multiplication of the original constraints, and by multiplication against sums of squares. That is, if $g_i(x) \geq 0$ and $g_j(x) \geq 0$ are valid inequalities, then so are $g_i(x)g_j(x) \geq 0$ and $g_i(x)s(x) \geq 0$, where $s(x)$ is SOS. After substitution with the extended variables λ , we then obtain a new system of linear equations and inequalities, with the property that the resulting inequality conditions are *semidefinite* conditions. The presence of the semidefinite constraints arises because we do not specify *a priori* what the multipliers $s(x)$ are, but only give their linear span.

EXAMPLE 12. Consider the polynomial system discussed earlier in Example 2. As described, new linear and semidefinite constraints are obtained by linearizing all the polynomial constraints in the original system. The corresponding relaxation is (for $d = 2$):

$$\begin{bmatrix} \lambda_1 & \lambda_{x_1} & \lambda_{x_2} \\ \lambda_{x_1} & \lambda_{x_1^2} & \lambda_{x_1 x_2} \\ \lambda_{x_2} & \lambda_{x_1 x_2} & \lambda_{x_2^2} \end{bmatrix} \succeq 0, \quad \lambda_{x_2} + \lambda_{x_1^2} + 2\lambda_1 = 0, \quad \lambda_{x_1} - \lambda_{x_2^2} + 3\lambda_1 \geq 0,$$

plus the condition $\lambda_1 > 0$ (without loss of generality, we can take $\lambda_1 = 1$). The first semidefinite constraint arises from linearizing the square of an arbitrary degree one polynomial, while the other two constraints are the direct linearization of the original equality and inequality constraints. The resulting problem is a semidefinite program, and in this case, its infeasibility directly shows that the original system of polynomial inequalities does not have a solution.

An appealing geometric interpretation follows from considering the projection of the feasible set of these relaxations in the space of original variables (i.e., λ_{x_i}). For the linear algebra relaxations of Section 2.2, we obtain outer approximations to the *affine hull* of the solution set (an algebraic variety), while the SDP relaxation described here constructs outer approximations to the *convex hull* of the corresponding semialgebraic set. This latter viewpoint will be discussed in Section 3.3, for the case of equations arising from combinatorial problems.

3.3. Theta bodies. Recall that traditional modeling of combinatorial optimization problems often uses 0/1 incidence vectors. The set S of solutions of a combinatorial problem (e.g., the stable sets, traveling salesman tours) is often computed through the (implicit) convex hull of such incidence vectors. Just as in the stable set and max-cut examples in Proposition 1.1, the incidence vectors can be seen at the set of *real* solutions to a system of polynomial equations: $f_1(x) = f_2(x) = \dots = f_m(x) = 0$, where $f_1, \dots, f_m \in \mathbb{R}[x] := \mathbb{R}[x_1, \dots, x_n]$. Over the years there have been well-known attempts to understand the structure of these convex hulls through semidefinite programming relaxations (see [47, 34, 26, 33]) and in fact they are closely related [27, 30]. Here we wish to summarize some recent results that give appealing structural properties, in terms of the associated system of equations (see [15, 14] for details).

Let us start with a historically important example: Given an undirected finite graph $G = (V, E)$, consider the set S_G of characteristic vectors of stable sets of G . The convex hull of S_G , denoted by $\text{STAB}(G)$, is the *stable set polytope*. As we mentioned already the vanishing ideal of S_G is given by $I_G := \langle x_i^2 - x_i \ (\forall i \in V), \ x_i x_j \ (\forall \{i, j\} \in E) \rangle$ which is a real radical zero-dimensional ideal in $\mathbb{R}[x]$. In [32], Lovász introduced a semidefinite relaxation, $\text{TH}(G)$, of the polytope $\text{STAB}(G)$, called the *theta body* of G . There are multiple descriptions of $\text{TH}(G)$, but the one in [34, Lemma 2.17], for instance, shows that $\text{TH}(G)$ can be defined completely in terms of the polynomial system I_G . It is easy to show that $\text{STAB}(G) \subseteq \text{TH}(G)$, and remarkably, we have that $\text{STAB}(G) = \text{TH}(G)$ if and only if the graph is *perfect*. We will now explain how the case of stable sets can be generalized to construct theta bodies for many other combinatorial problems.

We will construct an approximation of the convex hull of a finite set of points S , denoted $\text{conv}(S)$, by a sequence of convex bodies recovered from “degree truncations” of the defining polynomial systems. In what follows I will be a radical polynomial ideal. A polynomial f is *non-negative* modulo I , written as $f \geq 0 \bmod I$, if $f(s) \geq 0$ for all $s \in \mathcal{V}_{\mathbb{R}}(I)$. More strongly, the polynomial f is a *sum of squares (sos)* mod I if there exists $h_j \in \mathbb{R}[x]$ such that $f \equiv \sum_{j=1}^t h_j^2 \bmod I$ for some t , or equivalently, $f - \sum_{j=1}^t h_j^2 \in I$. If, in addition, each h_j has degree at most k , then we say that f is k -*sos* mod I . The ideal I is k -*sos* if every polynomial that is non-negative mod I is k -sos mod I . If every polynomial of degree at most d that is non-negative mod I is k -sos mod I , we say that I is (d, k) -*sos*.

Note that $\text{conv}(\mathcal{V}_{\mathbb{R}}(I))$, the convex hull of $\mathcal{V}_{\mathbb{R}}(I)$, is described by the linear polynomials f such that $f \geq 0 \bmod I$. A certificate for the non-negativity of $f \bmod I$ is the existence of a sos-polynomial $\sum_{j=1}^t h_j^2$ that is congruent to $f \bmod I$. One can now investigate the convex hull of S through the hierarchy of nested closed convex sets defined by the semidefinite programming relaxations of the set of $(1, k)$ -sos polynomials.

DEFINITION 3.1. Let $I \subseteq \mathbb{R}[x]$ be an ideal, and let k be a positive integer. Let $\Sigma_k \subset \mathbb{R}[x]$ be the set of all polynomials that are k -**sos** mod I .

1. The k -th theta body of I is

$$\text{TH}_k(I) := \{x \in \mathbb{R}^n : f(x) \geq 0 \text{ for every linear } f \in \Sigma_k\}.$$

2. The ideal I is TH_k -exact if the k -th theta body $\text{TH}_k(I)$ coincides with the closure of $\text{conv}(\mathcal{V}_{\mathbb{R}}(I))$.
3. The theta-rank of I is the smallest k such that $\text{TH}_k(I)$ coincides with the closure of $\text{conv}(\mathcal{V}_{\mathbb{R}}(I))$.

EXAMPLE 13. Consider the ideal $I = \langle x^2y - 1 \rangle \subset \mathbb{R}[x, y]$. Then $\text{conv}(\mathcal{V}_{\mathbb{R}}(I)) = \{(p_1, p_2) \in \mathbb{R}^2 : p_2 > 0\}$, and any linear polynomial that is non-negative over $\mathcal{V}_{\mathbb{R}}(I)$ is of the form $\alpha + \beta y$, where $\alpha, \beta \geq 0$. Since $\alpha y + \beta \equiv (\sqrt{\alpha}xy)^2 + (\sqrt{\beta})^2 \pmod{I}$, I is $(1, 2)$ -sos and TH_2 -exact.

EXAMPLE 14. For the case of the stable sets of a graph G , one can see that

$$\text{TH}_1(I_G) = \left\{ y \in \mathbb{R}^n : \begin{array}{l} \exists M \succeq 0, M \in \mathbb{R}^{(n+1) \times (n+1)} \text{ such that} \\ M_{00} = 1, \\ M_{0i} = M_{i0} = M_{ii} = y_i \forall i \in V \\ M_{ij} = 0 \forall \{i, j\} \in E \end{array} \right\}.$$

It is known that $\text{TH}_1(I_G)$ is precisely Lovász's theta body of G . The ideal I_G is TH_1 -exact precisely when the graph G is perfect.

By definition, $\text{TH}_1(I) \supseteq \text{TH}_2(I) \supseteq \dots \supseteq \text{conv}(\mathcal{V}_{\mathbb{R}}(I))$. As seen in Example 13, $\text{conv}(\mathcal{V}_{\mathbb{R}}(I))$ may not always be closed and so the theta-body sequence of I can converge, if at all, only to the closure of $\text{conv}(\mathcal{V}_{\mathbb{R}}(I))$. But the good news for combinatorial optimization is that there is plenty of good behavior for problems arising with a finite set of possible solutions.

3.4. Application: cuts and exact finite sets. We discuss now a few important combinatorial examples. As we have seen in Section 2.5 for 3-colorability, and in the preceding section for stable sets, in some special cases it is possible to give nice combinatorial characterizations of when low-degree certificates can exactly recognize infeasibility. Here are a few additional results for the real case:

EXAMPLE 15. For the max-cut problem we saw earlier, the defining vanishing ideal is $I(SG) = \langle x_e^2 - x_e \forall e \in E, x^T \forall T \text{ an odd cycle in } G \rangle$. In this case one can prove that the ideal $I(SG)$ is TH_1 -exact if and only if G is a bipartite graph. In general the theta-rank of $I(SG)$ is bounded above by the size of the max-cut in G . There is no constant k such that $\text{TH}_k(I(SG)) = \text{conv}(SG)$, for all graphs G . Other formulations of max-cut are studied in [14].

Recall that when $S \subset \mathbb{R}^n$ is a finite set, its vanishing ideal $I(S)$ is zero-dimensional and real radical (see [36] Section 12.5 for a definition of the real radical). In what follows, we say that a finite set $S \subset \mathbb{R}^n$ is *exact* if its vanishing ideal $I(S) \subseteq \mathbb{R}[x]$ is TH_1 -exact.

THEOREM 3.2 ([15]). For a finite set $S \subset \mathbb{R}^n$, the following are equivalent.

1. S is exact.
2. There is a finite linear inequality description of $\text{conv}(S)$ in which for every inequality $g(x) \geq 0$, g is 1-sos mod $I(S)$.
3. There is a finite linear inequality description of $\text{conv}(S)$ such that for every inequality $g(x) \geq 0$, every point in S lies either on the hyperplane $g(x) = 0$ or on a unique parallel translate of it.
4. The polytope $\text{conv}(S)$ is affinely equivalent to a compressed lattice polytope (every reverse lexicographic triangulation of the polytope is unimodular with respect to the defining lattice).

EXAMPLE 16. The vertices of the following 0/1-polytopes in \mathbb{R}^n are exact for every n : (1) hypercubes, (2) (regular) cross polytopes, (3) hyper-simplices (includes simplices), (4) joins of 2-level polytopes, and (5) stable set polytopes of perfect graphs on n vertices.

More strongly one can say the following.

PROPOSITION 3.1. Suppose $S \subseteq \mathbb{R}^n$ is a finite point set such that for each facet F of $\text{conv}(S)$ there is a hyperplane H_F such that $H_F \cap \text{conv}(S) = F$ and S is contained in at most $t + 1$ parallel translates of H_F . Then $I(S)$ is TH_t -exact.

In [15] the authors show that theta bodies can be computed explicitly as projections to the feasible set of a semidefinite program. These SDPs are constructed using the *combinatorial moment matrices* introduced by [29].

4. Recovering solutions in the feasible case. In principle, it is possible to find the actual roots of the system of equations (and thus the colorings, stable sets, or desired combinatorial object) whenever the relaxations are feasible and a few additional conditions are satisfied. Here we outline the linear algebra relaxations case, but the semidefinite case is very similar; see e.g. [18, 25] for this case.

We describe below how, under certain conditions, it is possible to recover the solution of the original polynomial system from the relaxations (linear or semidefinite) described in earlier sections. The main concepts are very similar for both methodologies, and are based on the well-known eigenvalue methods for polynomial equations; see e.g. [6, §2.4]. The key idea for extracting solutions is the fact that from the relaxations one can obtain a finite-dimensional representation of the vector space $\mathbb{K}[x]/I$ and its multiplicative structure, where I is the ideal $\mathbf{ideal}(F)$ (in the case of linear relaxations). In order to do this, we need to compute a basis of the vector space $\mathbb{K}[x]/I$, and construct matrix representations for the multiplication operators $M_{x_i} : \mathbb{K}[x]/I \rightarrow \mathbb{K}[x]/I$ where $[f] \mapsto [x_i f]$ for all $[f] \in \mathbb{K}[x]/I$. Then, we can use the eigenvalue/eigenvector methods to compute solutions (see e.g., [10]).

A sufficient condition for the existence of a suitable basis of $\mathbb{K}[x]/I$ is given by Theorem 2.2. Under this condition, multiplication matrices M_{x_i} can be easily computed. In particular, if we have computed a set $F \subset \mathbb{K}[x]$ that satisfies the conditions of Theorem 2.2 by running FPNullA, then

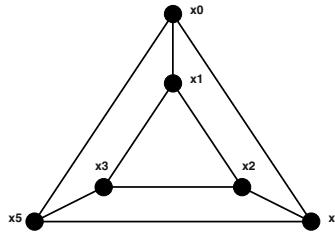


FIG. 5. Graph for Example 17.

finding a basis of R/I and computing its multiplicative structure is straightforward using linear algebra (see e.g., [37]). By construction, the matrices M_{x_i} commute pairwise, and to obtain the roots one must diagonalize the corresponding commutative algebra. It is well-known (see, e.g., [6]), that this can be achieved by forming a random linear combination of these matrices. This random matrix will generically have distinct eigenvalues, and the corresponding matrix of eigenvectors will give the needed change of basis. In the case of a finite field, it is enough to choose the random coefficients over an algebraic extension of sufficiently large degree, instead of working over the algebraic closure (alternatively, the more efficient methods in [11] can be used). The entries of the diagonalized matrices directly provide the coordinates of the roots.

REMARK 4.1. The condition in Theorem (2.2) can in general be a strong requirement for recovery of solutions, since it implies that we can obtain *all* solutions of the polynomial system. In some occasions, it may be desirable to obtain just a single solution, in which case weaker conditions may be of interest.

EXAMPLE 17. Consider the following polynomial system over \mathbb{F}_2 , that corresponds to the 3-colorings of the six-node graph in Figure 5:

$$x_i^3 + 1 = 0 \quad \forall i \in V, \quad x_i^2 + x_i x_j + x_j^2 = 0 \quad \forall (i, j) \in E.$$

We add to these equations the symmetry-breaking constraint $x_0 = 1$. After running NullA with this system as an input, we obtain multiplication matrices over \mathbb{F}_2 , of dimensions 4×4 , given by:

$$M_{x_1} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad M_{x_2} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad M_{x_3} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

$$M_{x_4} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad M_{x_5} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Diagonalizing the corresponding commutative algebra, we obtain the change of basis matrix given by

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \omega^2 & \omega & \omega & \omega^2 \\ 1 & 1 & \omega^2 & \omega \\ \omega^2 & \omega & 1 & 1 \end{bmatrix},$$

where ω is a primitive root of 1, i.e., it satisfies $\omega^2 + \omega + 1 = 0$. It can be easily verified that all the matrices $T^{-1}M_{x_i}T$ are diagonal, and given by:

$$\begin{aligned} T^{-1}M_{x_1}T &= \text{diag}[\omega, \omega^2, \omega, \omega^2] & T^{-1}M_{x_2}T &= \text{diag}[\omega^2, \omega, 1, 1] \\ T^{-1}M_{x_3}T &= \text{diag}[1, 1, \omega^2, \omega] & T^{-1}M_{x_4}T &= \text{diag}[\omega, \omega^2, \omega^2, \omega] \\ T^{-1}M_{x_5}T &= \text{diag}[\omega^2, \omega, \omega, \omega^2], \end{aligned}$$

which correspond to the four possible 3-colorings of the graph. For instance, from the second diagonal entry of each matrix we obtain the feasible coloring $(x_0, x_1, x_2, x_3, x_4, x_5) \rightarrow (1, \omega^2, \omega, 1, \omega^2, \omega)$.

Acknowledgements. We are grateful to the two anonymous referees who provided many useful corrections and comments that greatly enhanced the quality of presentation. We are also grateful to Jon Lee, Susan Margulies, Mohamed Omar, and Chris Hillar for their ideas and support.

APPENDIX

A. Proofs. This appendix contains proofs of some the results used in the main body of the paper that are either hard to find or whose original proof, available elsewhere, is not written in the language of this survey.

For the purpose of formally prove Theorem 2.2 we need to formalize further some of the notions of Section 2: The space $\mathbb{K}[[x]]$ is isomorphic to the dual vector space of $\mathbb{K}[x]$ consisting of all linear functionals on $\mathbb{K}[x]$, that is, $\mathbb{K}[[x]] \cong \text{Hom}(\mathbb{K}[x], \mathbb{K})$. We choose to use $\mathbb{K}[[x]]$ instead of as the dual vector space of $\mathbb{K}[x]$ (see e.g., [24]) because using $\mathbb{K}[[x]]$ makes clearer the linearization of a system of polynomial equations. The map $\tau : \mathbb{K}[[x]] \rightarrow \text{Hom}(\mathbb{K}[x], \mathbb{K})$ where $(\tau(\lambda))(g) = \sum_{\alpha \in \mathbb{N}^n} \lambda_\alpha g_\alpha = \lambda * g$ for all $\lambda \in \mathbb{K}[[x]]$ and $g \in \mathbb{K}[x]$ is an isomorphism and the inverse map is $\tau^{-1}(\psi) = \sum_{\alpha \in \mathbb{N}^n} \psi(x^\alpha)x^\alpha$ for all $\psi \in \text{Hom}(\mathbb{K}[x], \mathbb{K})$. For a given set $F \subseteq \mathbb{K}[x]$, there is an analogue of the annihilator F° in the context of the dual vector space $\text{Hom}(\mathbb{K}[x], \mathbb{K})$ as follows: $\text{Ann}(\mathbb{K}[x], F) := \{\psi \in \text{Hom}(\mathbb{K}[x], \mathbb{K}) : \psi(f) = 0, \forall f \in F\}$. Note that $F^\circ \cong \text{Ann}(\mathbb{K}[x], F)$ since $\tau(F^\circ) = \text{Ann}(\mathbb{K}[x], F)$.

LEMMA A.1. *Let $F \subseteq \mathbb{K}[x]$ be a vector subspace and $k \in \mathbb{N}$. Then, $\dim(\mathbb{K}[x]_k / (F \cap \mathbb{K}[x]_k)) = \dim_k(F^\circ)$.*

Proof. We know from Theorem 3.14 in [45], $\text{Ann}(\mathbb{K}[x], F \cap \mathbb{K}[x]_k) = \text{Ann}(\mathbb{K}[x], F) + \text{Ann}(\mathbb{K}[x], \mathbb{K}[x]_k)$; thus, $(F \cap \mathbb{K}[x]_k)^\circ = F^\circ + \mathbb{K}[x]_k^\circ$, and

so, we have $\pi_k((F \cap \mathbb{K}[x]_k)^\circ) = \pi_k(F^\circ) + \pi_k(\mathbb{K}[x]_k^\circ) = \pi_k(F^\circ)$. Moreover, from Theorems 3.12 and 3.15 in [45], we have $\text{Ann}(\mathbb{K}[x]_k, F \cap \mathbb{K}[x]_k) \cong \text{Hom}(\mathbb{K}[x]_k/(F \cap \mathbb{K}[x]_k), \mathbb{K}) \cong \mathbb{K}[x]_k/(F \cap \mathbb{K}[x]_k)$ since $\mathbb{K}[x]_k/(F \cap \mathbb{K}[x]_k)$ is finite dimensional, and finally, $\text{Ann}(\mathbb{K}[x]_k, F \cap \mathbb{K}[x]_k) \cong \pi_k((F \cap \mathbb{K}[x]_k)^\circ)$ since, for the isomorphism $\tau_k : \text{Hom}(\mathbb{K}[x]_k, \mathbb{K}) \rightarrow \mathbb{K}[x]_k$ where $\tau(\psi) = \sum_{\alpha \in \mathbb{N}^n : |\alpha| \leq k} \psi(x^\alpha)x^\alpha$, thus $\tau(\text{Ann}(\mathbb{K}[x]_k, F \cap \mathbb{K}[x]_k)) = \pi_k((F \cap \mathbb{K}[x]_k)^\circ)$. \square

We now present proofs verifying the correctness and efficiency of Algorithm 2. We begin by proving Theorem 2.2.

Proof. [Proof of Theorem 2.2] We will explicitly show that, under the hypothesis of the theorem, one can recover a finite dimensional vector space B such that $B \oplus F = \mathbb{K}[x]_d$ and $B \oplus I = \mathbb{K}[x]$. The result then follows from the equalities $\dim(I^\circ) = \dim(\mathbb{K}[x]/I) = \dim(B) = \dim(\mathbb{K}[x]_d/F) = \dim_d(F^\circ)$. We define the vector space $B \subseteq \mathbb{K}[x]_{d-1}$ such that $B \oplus (F \cap \mathbb{K}[x]_{d-1}) = \mathbb{K}[x]_{d-1}$. By assumption, $\dim_d(F^\circ) = \dim_{d-1}(F^\circ)$ implying $\dim(B) = \dim(\mathbb{K}[x]_{d-1}/F \cap \mathbb{K}[x]_{d-1}) = \dim(\mathbb{K}[x]_d/F)$, and thus, it follows that $B \oplus F = \mathbb{K}[x]_d$. It only remains to show that $B \oplus I = \mathbb{K}[x]$.

Denote $F^{[0]} = F$ and $F^{[k]} = (F^{[k-1]})^+$ for all $k \geq 1$. We show by induction on k that $B \oplus F^{[k]} = \mathbb{K}[x]_{d+k}$ for all $k \geq 0$, and hence $B \oplus I = \mathbb{K}[x]$. We have already established $B \oplus F = \mathbb{K}[x]_d$, so the claim holds for $k = 0$. The claim also holds for $k = 1$ as follows: $\mathbb{K}[x]_{d+1} = (\mathbb{K}[x]_d)^+ = (B \oplus F)^+ = B^+ + F^+ = B + F^+$ since $B^+ \subseteq \mathbb{K}[x]_d = B \oplus F$, and furthermore, the assumption $F^+ \cap \mathbb{K}[x]_d = F$ implies $F^+ \cap B = \emptyset$, and therefore, $B \oplus F^+ = \mathbb{K}[x]_{d+1}$. Now assume that the claim holds for $k \geq 1$ and let us prove it must hold for $k + 1$.

By the assumption that $B \oplus F^{[k]} = \mathbb{K}[x]_{d+k}$, there exists a vector space projection $\rho_k : \mathbb{K}[x]_{d+k} \rightarrow \mathbb{K}[x]_{d+k}$ where $\text{im}(\rho_k) = B$, $\rho_k(b) = b$ for all $b \in B$ and $\ker(\rho_k) = F^{[k]}$. We extend the map ρ_k to the map $\rho_{k+1} : \mathbb{K}[x]_{d+k+1} \rightarrow \mathbb{K}[x]_{d+k+1}$ by defining $\rho_{k+1}(g) := \rho_k(g_0) + \sum_i \rho_k(x_i \rho_k(g_i))$ where $g = g_0 + \sum_i x_i g_i$ is a representation of g with $g_0, g_1, \dots, g_n \in \mathbb{K}[x]_{d+k}$. We show below that ρ_{k+1} is well-defined meaning that the value of $\rho_{k+1}(g)$ is independent of the chosen representation of g since there may be multiple possible representations of g . It follows by construction that ρ_{k+1} is \mathbb{K} -linear, $\text{im}(\rho_{k+1}) = B$, $\rho_{k+1}(b) = b$ for all $b \in B$ and $\ker(\rho_{k+1}) = F^{[k+1]}$, implying that ρ_{k+1} is a vector space projection and $B \oplus F^{[k+1]} = \mathbb{K}[x]_{d+k+1}$ as required.

We now show that ρ_{k+1} is well-defined. First, consider the special case where $g \in \mathbb{K}[x]_{d+k+1}$ is a monomial, that is, $g = x_i x_j x^\gamma$ for some i, j and some monomial $x^\gamma \in \mathbb{K}[x]_{d+k-1}$, so $\rho_{k+1}(g) = \rho_k(x_i \rho_k(x_j x^\gamma))$ or $\rho_{k+1}(g) = \rho(x_j \rho_k(x_i x^\gamma))$. We thus need to show that $\rho_k(x_i \rho_k(x_j x^\gamma)) = \rho_k(x_j \rho_k(x_i x^\gamma))$. Now, $x^\gamma = b + f$ for some $b \in B$ where $\rho_k(x^\gamma) = b$ and $f \in F^{[k-1]}$ ($k \geq 1$). Then, $\rho_k(x_i x^\gamma) = \rho_k(x_i b + x_i f) = \rho_k(x_i b) + \rho_k(x_i f) = \rho_k(x_i b)$, and similarly, $\rho_k(x_j x^\gamma) = \rho_k(x_j b)$. Then,

$$\begin{aligned}
& \rho_k(x_i \rho_k(x_j x^\gamma)) - \rho_k(x_j \rho_k(x_i x^\gamma)) \\
&= \rho_k(x_i \rho_k(x_j b)) - \rho_k(x_j \rho_k(x_i b)) \\
&= \rho_k(x_i(x_j b - f_1)) - \rho_k(x_j(x_i b - f'_1)) && (f_1, f'_1 \in F) \\
&= (x_i(x_j b - f_1) - f_2) - (x_j(x_i b - f'_1) - f'_2) && (f_2, f'_2 \in F) \\
&= x_j f'_1 - x_i f_1 + f'_2 - f_2 \in F^+.
\end{aligned}$$

So, $\rho_k(x_i \rho_k(x^\alpha)) - \rho_k(x_j \rho_k(x^\beta)) \in F^+$. But, $\rho_k(x_i \rho_k(x^\alpha)) \in B$ and $\rho_k(x_j \rho_k(x^\beta)) \in B$ by definition, so $\rho_k(x_i \rho_k(x^\alpha)) - \rho_k(x_j \rho_k(x^\beta)) \in F^+ \cap B = \{0\}$ since $F^+ \cap \mathbb{K}[x]_d = F$. Thus, $\rho_k(x_i \rho_k(x^\alpha)) = \rho_k(x_j \rho_k(x^\beta))$ as required. By the \mathbb{K} -linearity of ρ_k , ρ_{k+1} is well-defined on $\mathbb{K}[x]_{d+k+1}$ as required. \square

Theorem 2.2 (and its proof) can be seen as an adaptation and simplification of Theorem 4.2 and Algorithm 4.3 in [37], the main difference being that in Mourrain's terminology, we stick to a particular *order ideal* and only need to keep track of vector space dimensions instead of an explicit basis for B .

We now present a proof of termination of the FPNuLA algorithm (see also the comments following Algorithm 4.3 in [37]).

LEMMA A.2. *Let I be a zero-dimensional ideal, then FPNuLA (Algorithm 2) terminates.*

Proof. First, we prove that the inner while loop must terminate. Let $F \subseteq \mathbb{K}[x]_d$ be a vector space. We denote $F^{[0,d]} = F$ and $F^{[k,d]} = (F^{[k-1,d]})^+ \cap \mathbb{K}[x]_d$ for all $k \geq 1$ where $d = \deg(F)$. By construction, $F^{[k,d]} \subseteq F^{[k+1,d]} \subseteq \mathbb{K}[x]_d$ for all k . So, the sequence of vector spaces $F^{[0,d]}, F^{[1,d]}, \dots, F^{[k,d]}, \dots$ is an inclusion-wise increasing sequence of vector subspaces of $\mathbb{K}[x]_d$. Since $\mathbb{K}[x]_d$ is finite-dimensional, the sequence must reach a fixed point where $F^{[k,d]} = F^{[k+1,d]}$, which is the terminating condition of the inner loop of FPNuLA (Steps 4-7). Let $F^{[*d]}$ denote this fixed point.

The outer loop of FPNuLA is essentially the same as NullA. After k iterations of the outer loop, the vector space F contains at least all linear combinations of polynomials of the form $x^\alpha f$ where the total degree $|\alpha| \leq k$ and where f is one of the initial polynomials in F . Therefore, if the system $F(x) = 0$ is infeasible, Hilbert's Nullstellensatz guarantees that after a finite number of iterations, $1 \in F$ and the algorithm terminates.

It remains to show that the algorithm terminates when the system $F(x) = 0$ is feasible. Let $I = I(F)$. Since I is zero-dimensional, there must exist a finite-dimensional vector space $B \subset \mathbb{K}[x]$ such that $\mathbb{K}[x] = I \oplus B$ (see e.g. [5, 50]). Since the system $F(x) = 0$ is feasible, Hilbert's Nullstellensatz implies $1 \notin I$. Thus, we can choose B such that $1 \in B$. Now after finitely many iterations of the outer loop, any $f \in I$ will eventually be in F . Combined with the fact that B^+ is finite dimensional and $B^+ \subset I \oplus B$, this implies that $B^+ \subset F \oplus B$ after finitely many iterations of the outer loop. Also, since the inner loop has terminated, we know that $F = F^+ \cap \mathbb{K}[x]_d = F^{[1,d]}$. Next, we show that $\mathbb{K}[x]_d = F \oplus B$. Now,

$(F \oplus B)^{[1,d]} = F^{[1,d]} + B^{[1,d]} = F \oplus B$ since $F = F^{[1,d]}$ and $B^+ \subseteq F \oplus B$. Thus, $(F \oplus B)^{[*,d]} = F \oplus B$, and since $B^+ \subseteq F \oplus B$, this implies

$$(B^+)^{[*,d]} \subseteq (F \oplus B)^{[*,d]} = F \oplus B.$$

But $1 \in B$, so $\mathbb{K}[x]_d \subseteq (B^+)^{[*,d]}$ which then implies $\mathbb{K}[x]_d = F \oplus B$. Then, since $B \subseteq \mathbb{K}[x]_{d-1}$, we also have $\mathbb{K}[x]_{d-1} = (F \cap \mathbb{K}[x]_{d-1}) \oplus B$, and thus, $\dim(\mathbb{K}[x]_d/F) = \dim(B) = \dim(\mathbb{K}[x]_{d-1}/F)$, which is the stopping criterion of the outer loop. \square

Now, we show that NullA and FPNullA algorithms run in polynomial time in the bit-size of the input data when the Nullstellensatz degree is assumed to be fixed. To begin, note that the number of monomials x^α with $\deg(x^\alpha) \leq k$ is $\binom{n+k}{k}$, which is $O(n^k)$.

Proof. (of Lemma 2.1). Let $d = \deg(F)$. First note that by definition (see section 2.1 in [46]) the input size of the defining basis $\{f_1, f_2, \dots, f_m\}$ of F equals $O(cm n^d)$ where c is the average bit-size of the coefficients in the basis.

For the proof of (1), observe that in the k^{th} iteration of Algorithm 1 (when the F^+ operation has increased the degree of F by k), we solve a system of linear equations $A_k x = b_k$ to find coefficients of the Nullstellensatz certificate in Step 2 of Algorithm 1. The rows of A_k consist of vectors of coefficients of all polynomials of the form $x^\alpha f_i$ where $i = 1, \dots, m$ and $\deg(x^\alpha) \leq k$. Therefore, A_k has $O(mn^k)$ rows and each row has input size $O(cn^{d+k})$. Hence, the input size of A_k is $O(cm n^{d+2k})$. The input size of b_k , which is a vector of zeros and ones, is $O(mn^k)$. Thus, the input size of the linear system $A_k x = b_k$ is $O(cm n^{d+2k})$, which is polynomial in the input size of the basis of F and n , and thus, the system can be solved in polynomial time (see e.g. Theorem 3.3 of [46]). The complexity of the first L iterations is thus bounded by L times the complexity of the L^{th} iteration, which is polynomial in L , n and the input size of the defining basis of F . This completes the proof of the first part.

We now prove part (2). Denote by F_k the vector space computed at the start of the k^{th} outer loop iteration. Let $\{g_1, \dots, g_{m_k}\}$ be a basis of F_k which was given to us either as an input or from the previous iteration. Observe that the $\deg(F_k) = d + k$, so each basis polynomial of F_k has bit size $O(\log_2(|\mathbb{K}|)n^{d+k})$. Note that $\dim(F_k) = m_k \leq O(n^{d+k})$; therefore the bit size of the entire basis $\{g_1, \dots, g_{m_k}\}$ is $M = O(\log_2(|\mathbb{K}|)n^{2(d+k)})$. Note that M is polynomial size in the input size of the initial basis f_1, \dots, f_m .

Now we proceed to analyze the cost of the k^{th} iteration, meaning steps 3 to 10 in the pseudocode. As in part (1), Step 3, involves solving a linear system of size M ; thus it can be done in polynomial time. In Step 4 we check whether $\dim(F_k) = \dim(F_k^+ \cap \mathbb{K}[x]_{d+k})$, which involves computing a basis of $F_k^+ \cap \mathbb{K}[x]_{d+k}$. Note that F_k^+ has bit size $(n+1)M$, and to compute the desired basis we perform Gaussian elimination on a matrix of size $(n+1)M$, which is polynomial time. If $\dim(F_k) \neq \dim(F_k^+ \cap \mathbb{K}[x]_{d+k})$,

then in Step 5, we set $F_k := F_k^+ \cap \mathbb{K}[x]_{d+k}$. We still have $F_k \subseteq \mathbb{K}[x]_d$, and F_k still has bit size M ; thus, as above, Step 6 can be computed in polynomial time. The number of iterations of the **while** loop (Steps 4-7) is $O(n^d)$ since the $\dim(F_k)$ is at most $\dim(\mathbb{K}[x]_d) = O(n^d)$ and $\dim(F_k)$ increases each iteration of the loop. So, the loop terminates in polynomial time. Then, Step 8 involves computing a basis for $F_k \cap \mathbb{K}[x]_{d-1}$ using Gaussian elimination, which is polynomial time again. Lastly, Step 9 involves computing a basis of F_k^+ which has bit size $(n+1)M$ and thus polynomial time. The complexity of the first L iterations is thus bounded by L times the complexity of the L th iteration, which is polynomial in $L, n, \log_2(|\mathbb{K}|)$ and the input size of the defining basis of F , and the result follows. \square

REFERENCES

- [1] E. BALAS, S. CERIA, AND G. CORNUÉJOLS, *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, Mathematical Programming, **58** (1993), pp. 295–324.
- [2] J. BOCHNAK, M. COSTE, AND M.-F. ROY, *Real algebraic geometry*, Springer, 1998.
- [3] M. CLEGG, J. EDMONDS, AND R. IMPAGLIAZZO, *Using the Groebner basis algorithm to find proofs of unsatisfiability*, in STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, New York, NY, USA, 1996, ACM, pp. 174–183.
- [4] N. COURTOIS, A. KLIMOV, J. PATARIN, AND A. SHAMIR, *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, in EUROCRYPT, 2000, pp. 392–407.
- [5] D. COX, J. LITTLE, AND D. O'SHEA, *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Springer Verlag, 1992.
- [6] ———, *Using Algebraic Geometry*, Vol. **185** of Graduate Texts in Mathematics, Springer, 2nd ed., 2005.
- [7] J. DE LOERA, C. HILLAR, P. MALKIN, AND M. OMAR, *Recognizing graph theoretic properties with polynomial ideals*. <http://arxiv.org/abs/1002.4435>, 2010.
- [8] J. DE LOERA, J. LEE, P. MALKIN, AND S. MARGULIES, *Hilbert's Nullstellensatz and an algorithm for proving combinatorial infeasibility*, in Proceedings of the Twenty-first International Symposium on Symbolic and Algebraic Computation (ISSAC 2008), 2008.
- [9] J. DE LOERA, J. LEE, S. MARGULIES, AND S. ONN, *Expressing combinatorial optimization problems by systems of polynomial equations and the nullstellensatz*, to appear in the Journal of Combinatorics, Probability and Computing (2008).
- [10] A. DICKENSTEIN AND I. EMIRIS, eds., *Solving Polynomial Equations: Foundations, Algorithms, and Applications*, Vol. **14** of Algorithms and Computation in Mathematics, Springer Verlag, Heidelberg, 2005.
- [11] W. EBERLY AND M. GIESBRECHT, *Efficient decomposition of associative algebras over finite fields*, Journal of Symbolic Computation, **29** (2000), pp. 441–458.
- [12] A.V. GELDER, *Another look at graph coloring via propositional satisfiability*, Discrete Appl. Math., **156** (2008), pp. 230–243.
- [13] E. GILBERT, *Random graphs*, Annals of Mathematical Statistics, **30** (1959), pp. 1141–1144.
- [14] J. GOUVEIA, M. LAURENT, P.A. PARRILO, AND R.R. THOMAS, *A new semidefinite programming relaxation for cycles in binary matroids and cuts in graphs*. <http://arxiv.org/abs/0907.4518>, 2009.

- [15] J. GOUVEIA, P.A. PARRILO, AND R.R. THOMAS, *Theta bodies for polynomial ideals*, SIAM Journal on Optimization, **20** (2010), pp. 2097–2118.
- [16] D. GRIGORIEV AND N. VOROBOJV, *Complexity of Nullstellensatz and Positivstellensatz proofs*, Annals of Pure and Applied Logic, **113** (2002), pp. 153–160.
- [17] D. HENRION AND J.-B. LASSERRE, *GloptiPoly: Global optimization over polynomials with MATLAB and SeDuMi*, ACM Trans. Math. Softw., **29** (2003), pp. 165–194.
- [18] ———, *Detecting global optimality and extracting solutions in GloptiPoly*, in Positive polynomials in control, Vol. **312** of Lecture Notes in Control and Inform. Sci., Springer, Berlin, 2005, pp. 293–310.
- [19] T. HOGG AND C. WILLIAMS, *The hardest constraint problems: a double phase transition*, Artif. Intell., **69** (1994), pp. 359–377.
- [20] A. KEHREIN AND M. KREUZER, *Characterizations of border bases*, Journal of Pure and Applied Algebra, **196** (2005), pp. 251 – 270.
- [21] A. KEHREIN, M. KREUZER, AND L. ROBBIANO, *An algebraist's view on border bases*, in Solving Polynomial Equations: Foundations, Algorithms, and Applications, A. Dickenstein and I. Emiris, eds., Vol. **14** of Algorithms and Computation in Mathematics, Springer Verlag, Heidelberg, 2005, ch. 4, pp. 160–202.
- [22] J. KOLLÁR, *Sharp effective Nullstellensatz*, Journal of the AMS, **1** (1988), pp. 963–975.
- [23] J. LASSERRE, *Global optimization with polynomials and the problem of moments*, SIAM J. on Optimization, **11** (2001), pp. 796–817.
- [24] J. LASSERRE, M. LAURENT, AND P. ROSTALSKI, *Semidefinite characterization and computation of zero-dimensional real radical ideals*, Found. Comput. Math., **8** (2008), pp. 607–647.
- [25] ———, *A unified approach to computing real and complex zeros of zero-dimensional ideals*, in Emerging Applications of Algebraic Geometry, M. Putinar and S. Sullivant, eds., vol. 149 of IMA Volumes in Mathematics and its Applications, Springer, 2009, pp. 125–155.
- [26] J.B. LASSERRE, *An explicit equivalent positive semidefinite program for nonlinear 0-1 programs*, SIAM J. on Optimization, **12** (2002), pp. 756–769.
- [27] M. LAURENT, *A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming*, Math. Oper. Res., **28** (2003), pp. 470–496.
- [28] ———, *Semidefinite relaxations for max-cut*, in The Sharpest Cut: The Impact of Manfred Padberg and His Work, M. Grötschel, ed., Vol. **4** of MPS-SIAM Series in Optimization, SIAM, 2004, pp. 257–290.
- [29] ———, *Semidefinite representations for finite varieties*, Mathematical Programming, **109** (2007), pp. 1–26.
- [30] ———, *Sums of squares, moment matrices and optimization over polynomials*, in Emerging Applications of Algebraic Geometry, M. Putinar and S. Sullivant, eds., Vol. **149** of IMA Volumes in Mathematics and its Applications, Springer, 2009, pp. 157–270.
- [31] J. LÖFBERG, *YALMIP: A toolbox for modeling and optimization in MATLAB*, in Proceedings of the CACSD Conference, Taipei, Taiwan, 2004.
- [32] L. LOVÁSZ, *Stable sets and polynomials*, Discrete Math., **124** (1994), pp. 137–153.
- [33] ———, *Semidefinite programs and combinatorial optimization*, in Recent advances in algorithms and combinatorics, B. Reed and C. Sales, eds., Vol. **11** of CMS Books in Mathematics, Spring, New York, 2003, pp. 137–194.
- [34] L. LOVÁSZ AND A. SCHRIJVER, *Cones of matrices and set-functions and 0-1 optimization*, SIAM J. Optim., **1** (1991), pp. 166–190.
- [35] S. MARGULIES, *Computer Algebra, Combinatorics, and Complexity: Hilbert's Nullstellensatz and NP-Complete Problems*, PhD thesis, UC Davis, 2008.
- [36] M. MARSHALL, *Positive polynomials and sums of squares*, Mathematical Surveys and Monographs, **146**. Providence, RI: American Mathematical Society (AMS). xii, p. 187, 2008.

- [37] B. MOURRAIN, *A new criterion for normal form algorithms*, in Proc. AAECC, Vol. **1719** of LNCS, Springer, 1999, pp. 430–443.
- [38] B. MOURRAIN AND P. TRÉBUCHET, *Stable normal forms for polynomial system solving*, Theoretical Computer Science, **409** (2008), pp. 229 – 240. Symbolic-Numerical Computations.
- [39] Y. NESTEROV, *Squared functional systems and optimization problems*, in High Performance Optimization, J.F. et al., eds., ed., Kluwer Academic, 2000, pp. 405–440.
- [40] P.A. PARRILLO, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*, PhD thesis, California Institute of Technology, May 2000.
- [41] ———, *Semidefinite programming relaxations for semialgebraic problems*, Mathematical Programming, **96** (2003), pp. 293–320.
- [42] P.A. PARRILLO AND B. STURMFELS, *Minimizing polynomial functions*, in Proceedings of the DIMACS Workshop on Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science (March 2001), S. Basu and L. Gonzalez-Vega, eds., American Mathematical Society, Providence RI, 2003, pp. 83–100.
- [43] S. PRAJNA, A. PAPACHRISTODOULOU, P. SEILER, AND P.A. PARRILLO, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2004.
- [44] G. REID AND L. ZHI, *Solving polynomial systems via symbolic-numeric reduction to geometric involutive form*, Journal of Symbolic Computation, **44** (2009), pp. 280–291.
- [45] S. ROMAN, *Advanced Linear Algebra*, Vol. **135** of Graduate Texts in Mathematics, Springer New York, third ed., 2008.
- [46] A. SCHRIJVER, *Theory of linear and integer programming*, Wiley, 1986.
- [47] H. SHERALI AND W. ADAMS, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM Journal on Discrete Mathematics, **3** (1990), pp. 411–430.
- [48] N.Z. SHOR, *Class of global minimum bounds of polynomial functions*, Cybernetics, **23** (1987), pp. 731–734.
- [49] G. STENGLE, *A Nullstellensatz and a Positivstellensatz in semialgebraic geometry*, Mathematische Annalen, **207** (1973), pp. 87–97.
- [50] H. STETTER, *Numerical Polynomial Algebra*, SIAM, 2004.
- [51] L. VANDENBERGHE AND S. BOYD, *Semidefinite programming*, SIAM Review, **38** (1996), pp. 49–95.
- [52] L. ZHANG, *zchaff v2007.3.12*. Available at <http://www.princeton.edu/~chaff/zchaff.html>, 2007.

MATRIX RELAXATIONS IN COMBINATORIAL OPTIMIZATION

FRANZ RENDL*

Abstract. The success of interior-point methods to solve semidefinite optimization problems (SDP) has spurred interest in SDP as a modeling tool in various mathematical fields, in particular in combinatorial optimization. SDP can be viewed as a matrix relaxation of the underlying 0-1 optimization problem. In this survey, some of the main techniques to get matrix relaxations of combinatorial optimization problems are presented. These are based either on semidefinite matrices, leading in general to tractable relaxations, or on completely positive or copositive matrices. Copositive programs are intractable, but they can be used to get exact formulations of many NP-hard combinatorial optimization problems. It is the purpose of this survey to show the potential of matrix relaxations.

Key words. Semidefinite optimization, Lift-and-project, integer programming.

1. Introduction. Integer programming and nonlinear optimization developed and grew rather independently of one another for a long time. The theoretical basis of integer programming consisted essentially of polyhedral combinatorics and the algorithmic machinery for linear programming, while nonlinear optimization relies on local analysis based on vector calculus (Taylor expansion, steepest descent principle, etc). In the last 15 years these two fields mutually opened up, and today interior-point methods are a widely accepted tool in integer programming, while the modeling power of 0-1 decision variables in an otherwise continuous setting expands the flexibility of real-world modeling substantially.

In this article we explore the idea of matrix liftings joining integer and nonlinear optimization. We first introduce the 0-1 formulation of an abstract combinatorial optimization problem (COP), given as follows. Let E be a finite set and let \mathcal{F} be a (finite) family of subsets of E . The elements $F \in \mathcal{F}$ represent the feasible solutions of (COP). Each $e \in E$ has a given integer cost c_e . We define the cost $c(F)$ of $F \in \mathcal{F}$ to be $c(F) := \sum_{e \in F} c_e$. The problem (COP) now consists in finding a feasible solution F of minimum cost:

$$(COP) \quad z^* = \min\{c(F) : F \in \mathcal{F}\}.$$

The 0-1 model of (COP) is obtained by assigning to each $F \in \mathcal{F}$ a characteristic vector $x_F \in \{0, 1\}^{|E|}$ with $(x_F)_e = 1$ if and only if $e \in F$. We can write (COP) as a linear program as follows. Let

$$\mathcal{P} := \text{conv}\{x_F : F \in \mathcal{F}\}$$

*Institut für Mathematik, Alpen-Adria-Universität Klagenfurt, Austria.

denote the convex hull of the characteristic vectors of feasible solutions. Then it is clear that

$$z^* = \min\{c^T x_F : F \in \mathcal{F}\} = \min\{c^T x : x \in P\}.$$

The first minimization is over the finite set \mathcal{F} , the second one is a linear program. This is the basic principle underlying the polyhedral approach to solve combinatorial optimization problems. The practical difficulty lies in the fact that in general the polyhedron \mathcal{P} is not easily available. We recall two classical examples to illustrate this point.

As a nice example, we consider first the **linear assignment problem**. For a given $n \times n$ matrix $C = (c_{ij})$, it consists of finding a permutation ϕ of $N = \{1, \dots, n\}$, such that $\sum_{i \in N} c_{i\phi(i)}$ is minimized. The set of all such permutations is denoted by Π . In our general setting, we define the ground set to be $E = N \times N$, the set of all ordered pairs (i, j) . Feasible solutions are now given through permutations ϕ as $F_\phi \subset E$ such that $F_\phi = \{(i, \phi(i)) : i \in N\}$. In this case the characteristic vector of F_ϕ is the permutation matrix X_ϕ given by $(X_\phi)_{ij} = 1$ if and only if $j = \phi(i)$. Birkhoff's theorem tells us that the convex hull of the set of permutation matrices Π is the set of doubly stochastic matrices $\Omega = \{X : Xe = X^T e = e, X \geq 0\}$ ¹.

THEOREM 1.1. $\text{conv}\{X_\phi : \phi \in \Pi\} = \Omega$.

Hence we have a simple polyhedral description of \mathcal{P} in this case. Therefore

$$\min \left\{ \sum_i c_{i\phi(i)} : \phi \in \Pi \right\} = \min\{\langle C, X \rangle : X \in \Omega\},$$

and the linear assignment problem can be solved as an ordinary linear program.

Unfortunately the set \mathcal{P} does not always have such a nice description. As a second example we consider the **stable set problem**. Given a graph $G = (V, E)$ with vertex set $V = \{1, \dots, n\}$, the problem is to find $S \subseteq V$, such that no edge joins vertices in S (such sets S are called stable), and $|S|$ is maximized. The ground set here is V and \mathcal{F} consists of all subsets of V which are stable. The characteristic vectors $x \in \mathbb{R}^n$ of the stable sets can be characterized by $x = (x_i)$ with $x_i \in \{0, 1\}$ and

$$x_i + x_j \leq 1 \quad \forall ij \in E(G), \tag{1.1}$$

because no stable set can contain both i and j if $ij \in E(G)$. A partial description of the convex hull of the characteristic vectors of stable sets is therefore given by

$$FSTAB(G) := \{x \in \mathbb{R}^n : x \geq 0, x_i + x_j \leq 1 \quad \forall ij \in E(G)\}, \tag{1.2}$$

¹For notation we refer to the end of this section.

leading to the following linear programming relaxation

$$\max\{e^T x : x \in FSTAB(G)\}.$$

If we take G to be the 5-cycle C_5 , we see that $x = \frac{1}{2}e$ is feasible for $FSTAB(C_5)$ with value $\frac{5}{2}$, showing that this is indeed only a relaxation of the stable set problem.

The use of a computationally tractable partial description of \mathcal{P} by linear inequalities in combination with systematic enumeration, like Branch and Bound, has led to quite successful solution methods for a variety of combinatorial optimization problems like the traveling salesman problem (TSP), see for instance [40]. It turned out however, that for some prominent NP-hard problems like Stable-Set or Max-Cut, this polyhedral approach was not as successful as one might have hoped in view of the results for TSP. It is the purpose of this article to describe matrix based relaxations, which generalize the purely polyhedral methods, and have the potential for stronger approximations of the original problem.

We conclude the introduction with a summary of the **notation** used throughout.

Vectors and matrices: e denotes the vector of all ones of appropriate dimension and $J = ee^T$ is the all ones matrix. $I = (e_1, \dots, e_n)$ denotes the $n \times n$ identity matrix, so the e_i 's represent the standard unit vectors. We also use $(\delta_{ij}) := I$, thereby defining the Kronecker delta δ_{ij} . The sum of the main diagonal entries of a square matrix A is called the trace, $tr(A) = \sum_i a_{ii}$. The inner product in \mathbb{R}^n as well as in the space of $n \times n$ matrices is represented by $\langle \cdot, \cdot \rangle$. Hence $\langle a, b \rangle = a^T b$, for $a, b \in \mathbb{R}^n$ and $\langle A, B \rangle = tr(A^T B)$ for matrices A, B . The Kronecker product of two matrices P, Q is the matrix consisting of all possible products of elements from P and Q , $P \otimes Q = (p_{ij}Q)$. For $m \in \mathbb{R}^n$ we define $Diag(m)$ to be the diagonal matrix having m on the main diagonal. $diag(M)$ is the vector, containing the main diagonal elements from M . If $X = (x_1, \dots, x_n)$ is a matrix with

columns x_i , then $\text{vec}(X) = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ is the vector, obtained by stacking the columns of X .

Sets and matrix cones: The standard simplex in \mathbb{R}^n is given by $\Delta = \{x \in \mathbb{R}^n : e^T x = 1, x \geq 0\}$. In this survey, we will put special emphasis on the following matrix cones.

$$\mathcal{N} := \{X : X \geq 0\} \text{ elementwise nonnegative matrices,}$$

$$\mathcal{S}^+ := \{X : X = X^T, a^T X a \geq 0 \ \forall a\}, \text{ positive semidefinite matrices,}$$

$$\mathcal{C} := \{X : X = X^T, a^T X a \geq 0 \ \forall a \geq 0\}, \text{ copositive matrices.}$$

We also use the notation $X \succeq 0$ to express $X \in \mathcal{S}^+$. If K is a cone in some finite dimensional vector space \mathbb{R}^d , then by definition the dual cone, denoted K^* , contains all elements from \mathbb{R}^d having nonnegative inner product with all elements from K ,

$$K^* := \{y \in \mathbb{R}^d : \langle x, y \rangle \geq 0 \ \forall x \in K\}.$$

It is well known and can easily be shown, that both \mathcal{N} and \mathcal{S}^+ are self-dual. The dual cone of \mathcal{C} is the cone

$$C^* = \text{conv}\{aa^T : a \geq 0\} = \{Y : \exists X \geq 0, Y = XX^T\}$$

of completely positive matrices. Membership in \mathcal{S}^+ can be checked in polynomial time, for instance through the existence (or non-existence) of the Cholesky decomposition. In contrast, it is NP-complete to decide if a matrix does not belong to \mathcal{C} , see [49]. While positive semidefinite matrices are covered in any reasonable text book on advanced linear algebra, we refer the reader to [4] for a thorough treatment of completely positive matrices and to [33] for a recent survey on copositive matrices.

Graphs: A graph $G = (V, E)$ is given through the set of vertices V and the set of edges E . We sometimes write $E(G)$ to indicate the dependence on G . If $S \subset V$, we denote by $\delta(S) := \{uv \in E : u \in S, v \notin S\}$ the set of edges joining S and $V \setminus S$. We also say that the edges in $\delta(S)$ are cut by S .

2. Matrix relaxations: basic ideas. The classical polyhedral approach is formulated as a relaxation in \mathbb{R}^n , the natural space to embed \mathcal{F} . Here n denotes the cardinality of E , $|E| = n$. Matrix-based relaxations of (COP) are easiest explained as follows. To an element $x_F \in \mathcal{F}$ we associate the matrix $x_F x_F^T$ and consider

$$\mathcal{M} := \text{conv}\{x_F x_F^T : F \in \mathcal{F}\}, \quad (2.1)$$

see for instance [44, 61, 62]. Note that

$$\text{diag}(x_F x_F^T) = x_F,$$

because $x_F \in \{0, 1\}^n$. This property immediately shows that the original linear relaxation, obtained through a partial description of \mathcal{P} can also be modeled in this setting. The full power of matrix lifting is based on the possibility to constrain \mathcal{M} to matrix cones other than polyhedral ones. Moreover, quadratic constraints on x_F will turn into linear constraints on matrices in \mathcal{M} .

If K is some matrix cone and matrices C, A_1, \dots, A_m and $b \in \mathbb{R}^m$ are given, the problem

$$\inf\{\langle C, X \rangle : \langle A_i, X \rangle = b_i \ i = 1, \dots, m, \ X \in K\} \quad (2.2)$$

is called a linear program over K . Linear programs over \mathcal{S}^+ are also called semidefinite programs (SDP) and those over C or C^* are called copositive

programs (CP) for short. In this paper we will mostly concentrate on SDP and CP relaxations of combinatorial optimization problems.

The duality theory of linear programming generalizes easily to conic linear programs. The (Lagrangian) dual associated to (2.2) is given as

$$\sup \{ b^T y : C - \sum_i y_i A_i \in K^* \}. \quad (2.3)$$

Weak duality ($\sup \leq \inf$) holds by construction of the dual. Strong duality ($\sup = \inf$), as well as attainment of the respective optima requires some sort of regularity of the feasible regions. We refer to Duffin [17] for the original paper, and to the handbook [70] on semidefinite programming for a detailed discussion of SDP. The existence of feasible points in the interior of the primal and dual cone insures the following characterization of optimality. For ease of notation we write $A(X) = b$ for the equations in (2.2). The linear operator A has an adjoint A^T , defined through the adjoint identity

$$\langle A(X), y \rangle = \langle X, A^T(y) \rangle.$$

We should point out that the inner product on the left is in \mathbb{R}^m and on the right it is in the space of $n \times n$ matrices. In this paper the inner products will always be canonical, so we do not bother to overload the notation to distinguish them. The adjoint can be expressed as $A^T(y) = \sum_i y_i A_i$.

THEOREM 2.1. [17, 70] *Suppose there exists $X_0 \in \text{int}(K)$ such that $A(X_0) = b$ and there is y_0 such that $C - A^T(y_0) \in \text{int}(K^*)$. Then the optima in (2.2) and (2.3) are attained. Moreover, X and y are optimal if and only if $A(X) = b$, $X \in K$, $Z := C - A^T(y) \in K^*$ and the optimal objective values coincide, $\langle X, Z \rangle = 0$.*

Matrix relaxations can be used in several ways to better understand (COP). The seminal work of Goemans and Williamson [23] opened the way to new approximation techniques for some COPs. We will briefly explain them as we develop the various relaxations. From a computational point of view, SDP based relaxations pose a serious challenge to existing algorithms for SDP. We will describe the currently most efficient ways to solve these relaxations (at least approximately).

There exist several recent survey papers devoted to the connection between semidefinite optimization and integer programming. The interested reader is referred to [39] for an extensive summary on the topic covering the development until 2003. The surveys by Lovász [43], Goemans [22] and Helmberg [29] all focus on the same topic, but also reflect the scientific interests and preferences of the respective authors. The present paper is no exception to this principle. The material selected, and also omitted, reflects the author's subjective view on the subject. It is a continuation and an extension of [57].

We are now going to look at several techniques to obtain matrix relaxations of combinatorial optimization problems. We start out with the generic idea, as explained in the introduction, and show how it works for graph partitioning.

3. Graph partition. Graph partition problems come in various formulations. The starting point is a graph G , given through its weighted $n \times n$ adjacency matrix A_G , or simply A . If $ij \in E(G)$, then a_{ij} denotes the weight of edge ij , otherwise $a_{ij} = 0$. Hence, $A = A^T$ and $\text{diag}(A) = 0$. The Laplacian L_A of A , or L for short, is defined to be the matrix

$$L := \text{diag}(Ae) - A. \quad (3.1)$$

The following simple properties of the Laplacian L will be used later on.

PROPOSITION 3.1. *The Laplacian L of the matrix A satisfies $Le = 0$ and $A \geq 0$ implies that $L \succeq 0$.*

Graph partition problems ask to separate the vertices of a graph into a specified number of partition blocks so that the total weight of edges joining different blocks is minimized or maximized. Partition problems lead rather naturally to matrix based relaxations because encoding whether or not vertices i and $j \in V$ are separated has a natural matrix representation, as we will see briefly. We recall the definition of a cut given by $S \subset V$: $\delta(S) = \{uv \in E(G) : u \in S, v \notin S\}$.

3.1. Max- k -Cut. For $k \geq 2$, Max- k -Cut asks to partition $V(G)$ into k subsets (S_1, \dots, S_k) such that the total weight of edges joining distinct subsets is maximized. We introduce characteristic vectors $s_i \in \{0, 1\}^n$ for each S_i . The $n \times k$ matrix $S = (s_1, \dots, s_k)$ is called the k -partition matrix. Since $\cup_i S_i = V$, we have

$$\sum_{i=1}^k s_i = Se = e.$$

Partition matrices have the following properties.

PROPOSITION 3.2. *Let $S = (s_1, \dots, s_k)$ be a k -partition matrix. Then $\text{diag}(SS^T) = e$, $kSS^T - J \succeq 0$.*

We prove a more general result, which will also be of use later on. Its proof has been pointed out by M. Laurent², see also [18], Lemma 2.

LEMMA 3.1. *Let s_1, \dots, s_k be a set of $0, 1$ vectors and $\lambda_i \geq 0$ be such that $\sum_{i=1}^k \lambda_i s_i = e$, hence $\sum_i \lambda_i = t > 0$. Let $M = \sum_i \lambda_i s_i s_i^T$. Then $\text{diag}(M) = e$, $tM - J \succeq 0$.*

Proof. Consider

$$\sum_i \lambda_i \begin{pmatrix} 1 \\ s_i \end{pmatrix} \begin{pmatrix} 1 \\ s_i \end{pmatrix}^T = \begin{pmatrix} t & e^T \\ e & M \end{pmatrix} \succeq 0.$$

²Oberwolfach conference on Discrete Optimization, November 2008.

We note that $\text{diag}(M) = \sum_i \lambda_i \text{diag}(s_i s_i^T) = \sum_i \lambda_i s_i = e$. The Schur-complement lemma shows that $M - \frac{1}{t}ee^T \succeq 0$. \square

Proposition 3.2 clearly follows with $\lambda_i = 1$. We recall that $\delta(S_i)$ denotes the set of edges joining S_i to $V \setminus S_i$. A simple calculation using basic properties of the Laplacian L shows that

$$s_i^T L s_i = \sum_{uv \in \delta(S_i)} a_{uv} \quad (3.2)$$

gives the weight of all edges cut by S_i . Therefore the total weight of all edges joining distinct subsets is given by

$$\frac{1}{2} \sum_i s_i^T L s_i = \frac{1}{2} \langle S, LS \rangle.$$

The factor $\frac{1}{2}$ comes from the fact that an edge $uv \in E(G)$ with $u \in S_i$ and $v \in S_j$ appears in both $\delta(S_i)$ and $\delta(S_j)$. Thus Max- k -Cut can be modeled as

$$\max \frac{1}{2} \langle S, LS \rangle$$

such that the $n \times k$ matrix S has entries 0 or 1 and $Se = e$. After replacing SS^T by Y , we get the following SDP relaxation

$$z_{GP-k} := \max \left\{ \frac{1}{2} \langle L, Y \rangle : \text{diag}(Y) = e, kY - J \in \mathcal{S}^+, Y \in \mathcal{N} \right\}. \quad (3.3)$$

The conditions $\text{diag}(Y) = e$ and $kY - J \in \mathcal{S}^+$ are derived from proposition 3.2. Note in particular that $Y \succeq 0$ is implied by $kY \succeq J \succeq 0$. The standard SDP formulation, see [19, 16], is obtained through the variable transformation

$$X = \frac{1}{k-1} [kY - J]$$

and yields, with $\langle J, L \rangle = 0$

$$\max \left\{ \frac{k-1}{2k} \langle L, X \rangle : \text{diag}(X) = e, X \in \mathcal{S}^+, x_{ij} \geq -\frac{1}{k-1} \right\}. \quad (3.4)$$

3.2. Max-Cut. The special case of Max- k -Cut with $k = 2$ is usually simply called Max-Cut, as the task is to separate V into S and $V \setminus S$ so as to maximize the weight of edges in $\delta(S)$. In view of (3.2) we clearly have

$$z_{MC} = \max \{ s^T L s : s \in \{0, 1\}^n \}.$$

Setting $y = e - 2s$, we have $y \in \{1, -1\}^n$ and, using $Le = 0$, we get

$$z_{MC} = \max \left\{ \frac{1}{4} y^T Ly : y \in \{-1, 1\}^n \right\}. \quad (3.5)$$

The following identity is a simple consequence of the definition of the Laplacian L through the adjacency matrix A , see (3.1)

$$\frac{1}{4}y^T Ly = \sum_{ij \in E(G)} a_{ij} \frac{1 - y_i y_j}{2}. \quad (3.6)$$

The resulting SDP relaxation becomes

$$\max\left\{\frac{1}{4}\langle L, X \rangle : \text{diag}(X) = e, X \in \mathcal{S}^+\right\}. \quad (3.7)$$

This model is identical to (3.4) with $k = 2$. We point out in particular that the sign constraint $x_{ij} \geq -1$ is implied by $X \succeq 0$ and $\text{diag}(X) = e$, and hence redundant.

3.3. k -Equicut. The following version of graph partition constrains the cardinalities of the partition blocks. In the simplest version of k -Equicut they are required to be equal to one another, $|S_i| = \frac{n}{k} \forall i$. Thus the column sums of S are $\frac{n}{k}$, $S^T e = \frac{n}{k}e$. We also have $Se = e$, because each vertex is in exactly one partition block. From this it follows that $SS^T e = \frac{n}{k}e$, so $\frac{n}{k}$ is the eigenvalue of SS^T for the eigenvector e . The relaxation from (3.3) therefore leads to

$$\min\left\{\frac{1}{2}\langle L, X \rangle : \text{diag}(X) = e, Xe = \frac{n}{k}e, X \geq 0, X \succeq 0\right\}.$$

In the context of Equicut, one is often interested in minimizing the total weight of edges cut. It is well known that without the cardinality constraints on the partition blocks, one can find the minimum cut (in the bisection case $k = 2$) using maximum flows. We also note that $X \succeq 0$ together with $Xe = \frac{n}{k}e$ implies $X = \frac{1}{k}J + \sum_i \lambda_i u_i u_i^T$ where the eigenvalues $\lambda_i \geq 0$ and the eigenvectors $u_i \perp e$. Therefore $kX - J \succeq 0$ as in (3.3) is implied. Further modeling ideas for Equicut using SDP can be found for instance in [36]. Applications of Equicut in telecommunication, and some computational experience with k -Equicut are shown in [41].

3.4. Approximation results for graph partition. The SDP relaxations for Max-Cut and Max- k -Cut can be used to get the following polynomial time approximations. The key idea underlying this approach was introduced by Goemans and Williamson [23] and consists of the following geometric construction. A feasible solution X of (3.4) or (3.7) has the Gram representation $X = (x_{ij})$ with $x_{ij} = (v_i^T v_j)$. The constraint $\text{diag}(X) = e$ implies that the v_i are unit vectors.

Let us first consider Max-Cut. Goemans and Williamson [23] interpret the Max-Cut problem as finding an embedding v_i of the vertices i in the unit sphere in \mathbb{R}^1 , hence $v_i \in \{-1, 1\}$, such that $\frac{1}{4} \sum_{ij} l_{ij} v_i^T v_j$ is maximized, see (3.5).

The optimal solution of the relaxation (3.7) gives such an embedding in \mathbb{R}^d , where d is the rank of an optimal X . Clearly $1 \leq d \leq n$. How should

we get a (bi)partition of approximately maximum weight? Goemans and Williamson propose the following simple but powerful hyperplane rounding trick.

Take a random hyperplane H through the origin, and let $S \subseteq V$ be the set of vertices on one side of H . The probability that H separates i and j is proportional to the angle between v_i and v_j and is given by

$$\frac{1}{\pi} \arccos(v_i^T v_j).$$

In [23] it is shown that

$$\frac{1}{\pi} \arccos t \geq \alpha \frac{1}{2}(1-t)$$

holds for $-1 \leq t \leq 1$ and $\alpha \approx 0.87856$. We therefore get the following performance bound for the expected value of the cut y obtained this way, provided $a_{ij} \geq 0$

$$\sum_{ij \in E(G)} a_{ij} \frac{1 - y_i y_j}{2} \geq \alpha \sum_{ij} a_{ij} \frac{1}{2}(1 - v_i^T v_j) \approx 0.87856 z_{MC}.$$

Note the use of (3.6). Later on, Nesterov [51] generalizes this result to the more general case where only $L \succeq 0$ is assumed. The analysis in this case shows that the expected value of the cut y obtained from hyperplane rounding is at least

$$\frac{1}{4} y^T L y \geq \frac{2}{\pi} z_{MC} \approx 0.636 z_{MC}.$$

Frieze and Jerrum [19] generalize the hyperplane rounding idea to Max- k -Cut. Starting again from the Gram representation $X = V^T V$ with unit vectors v_i forming V , we now take k independent random vectors $r_1, \dots, r_k \in \mathbb{R}^n$ for rounding. The idea is that partition block S_h contains those vertices i which have v_i most parallel to r_h ,

$$i \in S_h \iff v_i^T r_h = \max\{v_i^T r_l : 1 \leq l \leq k\}.$$

Ties are broken arbitrarily. For the computation of the probability that two vertices are in the same partition block, it is useful to assume that the entries of the r_i are drawn independently from the standard normal distribution

$$\Pr(v_s, v_t \in S_1) = \Pr(v_s^T r_1 = \max_i v_s^T r_i, v_t^T r_1 = \max_i v_t^T r_i).$$

The symmetry properties of the normal distribution imply that this probability depends on $\rho = \cos(v_s^T v_t)$ only. We denote the resulting probability by $I(\rho)$. Therefore

$$\Pr(v_s \text{ and } v_t \text{ not separated}) = kI(\rho).$$

The computation of $I(\rho)$ involves multiple integrals. A Taylor series expansion is used in [19] to get the following estimates for the expectation value of the cut given by the partition S from hyperplane rounding,

$$\frac{1}{2}\langle S, LS \rangle \geq \alpha_k z_{GP-k},$$

where $\alpha_2 = 0.87856$ as for Max-Cut, $\alpha_3 \approx 0.8327$, $\alpha_4 \approx 0.85$. In [19], values for α_k are also provided for larger values of k . Later, these bounds on α_k were slightly improved, see [16]. It should be emphasized that the mathematical analysis underlying this simple rounding scheme involves rather subtle techniques from classical calculus to deal with probability estimates leading to the final error bounds α_k , see also the monograph [14].

4. Stable sets, cliques and coloring. The seminal work of Lovász [42] introduces a semidefinite program, which can be interpreted both as a relaxation of the Max-Clique problem and the Coloring problem.

4.1. Stable sets and Cliques. The Stable-Set problem has already been described in the introduction. We denote by $\alpha(G)$ the stability number of G (= cardinality of the largest stable set in G). It is given as the optimal solution of the following integer program.

$$\alpha(G) = \max\{e^T x : x \in \{0, 1\}^n, x_i + x_j \leq 1 \forall ij \in E(G)\}.$$

We first observe that the inequalities could equivalently be replaced by

$$x_i x_j = 0 \forall ij \in E(G).$$

A clique in G is a subset of pairwise adjacent vertices. Moving from G to the complement graph \overline{G} , which joins vertices $i \neq j$ whenever $ij \notin E(G)$, it is immediately clear that stable sets in G are cliques in \overline{G} and vice-versa.

In the spirit of matrix lifting, we introduce for a nonzero characteristic vector x of some stable set the matrix

$$X := \frac{1}{x^T x} xx^T. \quad (4.1)$$

These matrices satisfy

$$X \succeq 0, \text{tr}(X) = 1, x_{ij} = 0 \forall ij \in E(G).$$

We also have $\langle J, X \rangle = \frac{(e^T x)^2}{x^T x} = e^T x$. We collect the equations $x_{ij} = 0 \forall ij \in E(G)$ in the operator equation $\mathcal{A}_G(X) = 0$. Therefore we get the semidefinite programming upper bound $\alpha(G) \leq \vartheta(G)$,

$$\vartheta(G) := \max\{\langle J, X \rangle : \text{tr}(X) = 1, \mathcal{A}_G(X) = 0, X \succeq 0\}. \quad (4.2)$$

This is in fact one of the first relaxations for a combinatorial optimization problem based on SDP. It was introduced by Lovász [42] in 1979. This

problem has been the starting point for many quite far reaching theoretical investigations. It is beyond the scope of this paper to explain them in detail, but here are some key results.

Grötschel, Lovász and Schrijver [25] show that $\alpha(G)$ can be computed in polynomial time for **perfect graphs**. This is essentially a consequence of the tractability to compute $\vartheta(G)$, and the fact that $\alpha(G) = \vartheta(G)$ holds for perfect graphs G . We do not explain the concept of perfect graphs here, but refer for instance to [25, 60]. It is however a prominent open problem to provide a polynomial time algorithm to compute $\alpha(G)$ for perfect graphs, which is purely combinatorial (=not making use of $\vartheta(G)$).

The Stable-Set problem provides a good example for approximations based on other matrix relaxations. Looking at (4.1), we can additionally ask that $X \in \mathcal{N}$. In this case the individual equations $x_{ij} = 0 \forall ij \in E(G)$ can be added into a single equation $\sum_{ij \in E(G)} x_{ij} = 0$. If we use A_G for the adjacency matrix of G , this means

$$\langle A_G, X \rangle = 0.$$

Hence we get the stronger relaxation, proposed independently by Schrijver [59] and McEliece et al [47].

$$\alpha(G) \leq \max\{\langle J, X \rangle : \langle A_G, X \rangle = 0, \text{tr}(X) = 1, X \in \mathcal{S}^+ \cap \mathcal{N}\} =: \vartheta'(G).$$

In terms of matrix cones, we have moved from \mathcal{S}^+ to $\mathcal{S}^+ \cap \mathcal{N}$. We can go one step further. The matrix X from (4.1) is in fact completely positive, hence we also get

$$\alpha(G) \leq \max\{\langle J, X \rangle : \langle A_G, X \rangle = 0, \text{tr}(X) = 1, X \in C^*\}. \quad (4.3)$$

We will see shortly that the optimal value of the copositive program on the right hand side is in fact equal to $\alpha(G)$. This result is implicitly contained in Bomze et al [8] and was stated explicitly by de Klerk and Pasechnik [15]. A simple derivation can be obtained from the following theorem of Motzkin and Straus [48]. We recall that $\Delta := \{x \in \mathbb{R}^n : x \geq 0, e^T x = 1\}$.

THEOREM 4.1. [48] *Let A_G be the adjacency matrix of a graph G . Then*

$$\frac{1}{\alpha(G)} = \min\{x^T(A_G + I)x : x \in \Delta\}. \quad (4.4)$$

The relation (4.4) implies in particular that

$$0 = \min\{x^T(A_G + I - \frac{1}{\alpha}ee^T)x : x \in \Delta\} = \min\{x^T(A_G + I - \frac{1}{\alpha}J)x : x \geq 0\}.$$

This clearly qualifies the matrix $\alpha(A_G + I) - J$ to be copositive. Therefore

$$\inf\{\lambda : \lambda(A_G + I) - J \in C\} \leq \alpha(G).$$

But weak duality for conic linear programs also shows that

$$\sup\{\langle J, X \rangle : \langle A_G + I, X \rangle = 1, X \in C^* \} \leq \inf\{\lambda : \lambda(A_G + I) - J \in C\}.$$

Finally, any matrix X of the form (4.1) is feasible for the sup-problem, hence

$$\alpha(G) \leq \sup\{\langle J, X \rangle : \langle A_G + I, X \rangle = 1, X \in C^* \}.$$

Combining the last 3 inequalities, we see that equality must hold throughout, the infimum is attained at $\lambda = \alpha(G)$ and the supremum is attained at (4.1) with x being a characteristic vector of a stable set of size $\alpha(G)$. Hence we have shown the following result.

THEOREM 4.2. [15] *Let G be a graph. Then*

$$\alpha(G) = \max\{\langle J, X \rangle : \langle A_G + I, X \rangle = 1, X \in C^* \}.$$

This shows on one hand that copositive programs are intractable. It also shows however, that models based on CP may be substantially stronger than SDP based models. We will see more of this in some of the subsequent sections.

4.2. Coloring. Partitioning the vertex set V of a graph into stable sets is pictorially also called **vertex coloring**. Each partition block S_i receives a distinct 'color', and vertices having the same color are non-adjacent, because S_i is stable. We could for instance partition into singletons, so each vertex would get a distinct color. The **chromatic number** $\chi(G)$ is the smallest number k such that G has a k -partition into stable partition blocks.

If we let $\{S_1, \dots\}$ denote the set of all stable sets of G , then the following integer program determines $\chi(G)$,

$$\chi(G) = \min\left\{\sum_i \lambda_i : \sum_i \lambda_i x_i = e, \lambda_i \in \{0, 1\}\right\}.$$

x_i denotes the characteristic vector of the stable set S_i . It should be observed that the number of variables λ_i is in general not polynomially bounded in $|V(G)|$. The **fractional chromatic number** $\chi_f(G)$ is obtained by allowing $\lambda_i \geq 0$,

$$\chi_f(G) = \min\left\{\sum_i \lambda_i : \sum_i \lambda_i x_i = e, \lambda_i \geq 0\right\}. \quad (4.5)$$

This is now a linear program with a possibly exponential number of variables. Computing $\chi_f(G)$ is known to be NP-hard, see for instance [60].

A semidefinite programming based lower bound on $\chi_f(G)$ can be obtained as follows, see Lovász [42] and Schrijver [60]. Let $\lambda_i \geq 0$ be an

optimal solution of (4.5), so $\chi_f(G) = \sum_i \lambda_i$. Since $\sum_i \lambda_i x_i = e$, we can apply Lemma 3.1. The matrix

$$M = \sum_i \lambda_i x_i x_i^T \quad (4.6)$$

therefore satisfies

$$\chi_f(G)M - J \in \mathcal{S}^+, \text{ diag}(M) = e.$$

Moreover, since each x_i is characteristic vector of a stable set in G , we also have $m_{uv} = 0$ $uv \in E(G)$, or $\mathcal{A}_G(M) = 0$. Therefore the optimal value of the following SDP is a lower bound on $\chi_f(G)$,

$$\chi_f(G) \geq \min\{t : \text{diag}(M) = e, \mathcal{A}_G(M) = 0, tM - J \succeq 0\}. \quad (4.7)$$

Strictly speaking, this is not a linear SDP, because both t and M are variables, but it can easily be linearized by introducing a new matrix variable Y for tM and asking that $\text{diag}(Y) = te$. The resulting problem is the dual of

$$\max\{\langle J, X \rangle : \langle I, X \rangle = 1, x_{ij} = 0 \text{ } ij \notin E(G), X \succeq 0\},$$

which is equal to $\vartheta(\overline{G})$. Thus we have shown the Lovász 'sandwich theorem'. In [42], the weaker upper bound $\chi(G)$ is shown for $\vartheta(\overline{G})$, but it is quite clear that the argument goes through also with $\chi_f(G)$.

THEOREM 4.3. [42] *Let G be a graph. Then $\alpha(\overline{G}) \leq \vartheta(\overline{G}) \leq \chi_f(G)$.*

Let us now imitate the steps leading from $\vartheta(G)$ to the copositive strengthening of the stability number from the previous section. The crucial observation is that M from (4.6) and therefore $tM \in C^*$.

This leads to the following conic problem, involving matrices both in \mathcal{S}^+ and C^*

$$t^* := \min\{t : \text{diag}(M) = e, \mathcal{A}_G(M) = 0, tM - J \in \mathcal{S}^+, M \in C^*\}.$$

Using again Lemma 3.1 we see that M from (4.6) is feasible for this problem, therefore

$$t^* \leq \chi_f(G).$$

In [18] it is in fact shown that equality holds.

THEOREM 4.4. [18] *The optimal value t^* of the above SDP-CP relaxation of the chromatic number is equal to $\chi_f(G)$.*

This shows again the strength of modeling with copositive programs. New relaxations of the chromatic number based on graph products have very recently been introduced in [26, 27, 28]. It is beyond the scope of this introductory survey to elaborate on this approach.

4.3. Approximation results for coloring. Similar to Max- k -Cut we can use the SDP relaxations of coloring to derive a vertex partition using hyperplane rounding. An additional complication comes from the fact that the partition blocks have to be stable sets. The first ground-breaking results were obtained by Karger et al [35]. We briefly explain some of the ideas for the case of graphs G having $\chi(G) = 3$. This may seem artificial, but simply knowing that $\chi(G) = 3$ does not help much. In fact, finding a 4-coloring in a 3-colorable graph is NP-hard, see [37].

Widgerson [68] observes that if the largest degree d_{\max} of a graph on n vertices is large, $d_{\max} > \sqrt{n}$, then 2 colors suffice to color the neighbourhood of the vertex with largest degree, thereby legally coloring at least \sqrt{n} vertices. If $d_{\max} \leq \sqrt{n}$, then the graph can be colored with $\sqrt{n} + 1$ colors. This yields a simple algorithm that colors any three-colorable graph with at most $3\sqrt{n}$ colors, see [68].

In [35], the SDP underlying $\vartheta(G)$ is used for hyperplane rounding. The key observation is that a carefully chosen rounding procedure can be used to produce with high probability a stable set of size $\tilde{O}(\frac{n}{d_{\max}^{1/3}})$. The \tilde{O} notation ignores polylogarithmic terms. This leads to a coloring with $\tilde{O}(n^{1/4})$ colors. Based on this approach, Blum and Karger [6] refine the analysis and end up with colorings using $\tilde{O}(n^{3/14})$ colors. Note that $3/14 \approx 0.2143$. Very recently, these results were further improved by Arora et al [2] to at most $\tilde{O}(n^{0.2111})$ colors. The derivation of these estimates is rather complex. We refer to the recent dissertation [13] for a detailed discussion of hyperplane rounding for coloring.

5. Bandwidth of graphs. The **minimum bandwidth problem** of a graph G can be interpreted as a reordering problem of the vertices of G such that the nonzero entries of the adjacency matrix (after reordering) are collected within a band of small width around the main diagonal. Formally we denote the vertices of G again by $V = \{1, \dots, n\}$. For a permutation $\phi \in \Pi$, the bandwidth $bw(\phi)$ of ϕ is defined as

$$bw(\phi) := \max\{|\phi(i) - \phi(j)| : ij \in E(G)\}.$$

The bandwidth of G is the smallest of these values over all bijections $\phi : V \mapsto V$.

$$bw(G) := \min\{bw(\phi) : \phi \in \Pi\}. \quad (5.1)$$

Determining $bw(G)$ is in general NP-hard, see [52]. It remains NP-hard, even if G is restricted to be a tree with maximum degree 3, see [45].

The bandwidth problem has many applications. Consider for instance a sparse symmetric system of linear equations. Having an ordering of the system matrix with small bandwidth may result in a substantial computational speedup, when actually solving the system.

In the following approximation approach, Blum et al [7] formulate the bandwidth problem as an ordering of V on n equidistant points on a quarter-circle of radius n . Let

$$P_j := n(\cos \frac{j\pi}{2n}, \sin \frac{j\pi}{2n}), \quad j = 1, \dots, n$$

denote these points. The problem

$$\min\{b : \exists \phi \in \Pi \text{ such that } v_i = P_{\phi(i)}, \|v_i - v_j\| \leq b \forall ij \in E(G)\}$$

is clearly equivalent to finding $bw(G)$. Blum et. al. relax the difficult part of bijectively mapping V to $\{P_1, \dots, P_n\}$. Let us define the constraints

$$\|v_i\| = n \quad \forall i \in V, \quad \|v_i - v_j\| \leq b \quad \forall ij \in E. \quad (5.2)$$

Simply solving

$$\min\{b : v_i \text{ satisfy (5.2)}\}$$

could be done using SDP by introducing $X = V^T V \succeq 0$. The constraints (5.2) translate into

$$x_{ii} = n^2 \quad \forall i, \quad x_{ii} + x_{jj} - 2x_{ij} \leq \beta \quad \forall ij \in E \quad (5.3)$$

and minimizing β would yield $b = \sqrt{\beta}$. Unfortunately, the optimum of this SDP has value $\beta = 0$, by assigning each i to the same point, say P_1 . To force some spreading of the v_i , we observe the following.

PROPOSITION 5.1. *Let $i \in V$ and $S \subseteq V \setminus i$. Then*

$$\sum_{j \in S} (i - j)^2 \geq \frac{|S|}{6}(|S| + 1)(\frac{|S|}{2} + 1) =: f(|S|).$$

Since $\|P_i - P_j\| \geq |i - j|$, we can include these additional spread constraints

$$\sum_{j \in S} \|v_i - v_j\|^2 \geq f(|S|) \quad \forall S \subseteq V, i \in V \quad (5.4)$$

into the SDP. Blum et. al. [7] consider the following strengthened problem, which is equivalent to an SDP, once we make the change of variables $X = V^T V$

$$\min\{b : (v_i) \text{ satisfy (5.2), (5.4)}\}. \quad (5.5)$$

Even though there is an exponential number of these spread constraints, it can easily be argued that their separation can be done in polynomial time by sorting, for i fixed, the remaining vertices $j \in V \setminus i$ in increasing order of $\|v_i - v_j\|^2$.

Having an optimal solution b, v_1, \dots, v_n of the SDP, Blum et al apply the hyperplane rounding idea as follows. Take a random line through the origin and project the v_i onto this line. The resulting permutation of the vertices is shown to satisfy the following estimate.

THEOREM 5.1. [7] *Let G be a graph and b, v_1, \dots, v_n denote the optimal solution of (5.5). The ordering ϕ produced by projecting the v_i onto a random line through the origin satisfies*

$$bw(\phi) \leq O(\sqrt{n/b} \log(n))bw(G)$$

with high probability.

The proof of this result is rather long and technical and is omitted here.

We are now going to describe another modeling approach which can be used to bound $bw(G)$. Let G be a connected graph, given through its adjacency matrix $A \geq 0$. We now consider 3-partitions (S_1, S_2, S_3) of V having prescribed cardinalities $|S_i| = m_i$. Let us denote the edges

joining S_1 and S_2 by $\delta(S_1, S_2)$. If we set $B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ and use

the characteristic vectors s_i for S_i , we get the total weight of edges in $\delta(S_1, S_2)$ by

$$s_1^T A s_2 = \frac{1}{2} \langle S, ASB \rangle.$$

Minimizing this cost function over 3-partitions of prescribed cardinalities $m = (m_1, m_2, m_3)$ may look artificial, but we will see that it provides a handle to several graph optimization problems, in particular the bandwidth problem. So let us consider

$$z_{3gp} := \min\left\{\frac{1}{2} \langle S, ASB \rangle : Se = e, S^T e = m, S \geq 0, S^T S = M\right\}. \quad (5.6)$$

We set $M = \text{Diag}(m)$. This is a nonconvex quadratic optimization problem, hence intractable. We leave it to the reader to verify that feasible matrices S of this problem are integer. We can use this problem as follows. If $z_{3gp} = 0$ then the graph underlying A has a **vertex separator** of size m_3 . (We recall that $S \subset V$ is a vertex separator, if the removal of S disconnects the graph.) If $z_{3gp} > 0$ then the bandwidth of A is at least $m_3 + 1$, see Helmberg et al [31]. This last condition could also be used if we would have a positive lower bound on z_{3gp} . In [31], such a lower bound was derived using eigenvalue techniques on the basis that the columns of S are pairwise orthogonal. Povh and Rendl [54] consider the matrix lifting

$$\mathcal{M}_{3gp} := \text{conv}\{ss^T : s = \text{vec}(S), S \text{ is 3-partition with } |S_i| = m_i\}. \quad (5.7)$$

It turns out that \mathcal{M}_{3gp} can be represented as the intersection of a set of linear equations with the cone C^* of completely positive matrices. To see

how these equations are derived, we start out with the quadratic equation $S^T S = M$. At this point the Kronecker product of two matrices P and Q , given by $P \otimes Q := (p_{ij}Q)$, is extremely useful. If P, Q and S have suitable size, and $s = \text{vec}(S)$, the following identity is easy to verify

$$\langle S, PSQ \rangle = \langle Q^T \otimes P, ss^T \rangle. \quad (5.8)$$

Using it, we see that $(S^T S)_{ij} = e_j^T S^T S e_i = \langle e_j e_i^T \otimes I, ss^T \rangle$. The symmetry of ss^T allows us to replace $e_j e_i^T$ by $B_{ij} := \frac{1}{2}(e_j e_i^T + e_i e_j^T)$. Therefore $(S^T S)_{ij} = M_{ij}$ becomes

$$\langle B_{ij} \otimes I, Y \rangle = M_{ij} \quad (5.9)$$

where $Y \in \mathcal{M}_{3gp}$. In a similar way we get

$$\langle J_3 \otimes e_i e_i^T, Y \rangle = 1, \quad 1 \leq i \leq n, \quad (5.10)$$

by squaring the n equations $Se = e$. Pairwise multiplication of the constraints $S^T e = m$ gives

$$\langle B_{ij} \otimes J_n, Y \rangle = m_i m_j \quad 1 \leq i \leq j \leq 3. \quad (5.11)$$

Finally, elementwise multiplication of $Se = e$ with $S^T e = m$ gives

$$\langle e_i e_i^T \otimes ee_j^T, Y \rangle = m_i \quad 1 \leq i \leq 3, \quad 1 \leq j \leq n. \quad (5.12)$$

The following result is shown in [54].

THEOREM 5.2.

$$\mathcal{M}_{3gp} = \{Y : Y \in C^*, \quad Y \text{ satisfies (5.9), (5.10), (5.11), (5.12)}\}.$$

Therefore z_{3gp} could be determined as the optimal solution of the (intractable) copositive program

$$z_{3gp} = \min\left\{\frac{1}{2}\langle B \otimes A, Y \rangle : \quad Y \in C^*, \quad Y \text{ satisfies (5.9), (5.10), (5.11), (5.12)}\right\}.$$

We get a tractable relaxation by asking $Y \in \mathcal{S}^+$ instead of $Y \in C^*$. Further details can be found in [54] and in the dissertation [53]. While the model investigated by [7] is based on $n \times n$ matrices, it has to be emphasized that the last model uses $3n \times 3n$ matrices, and hence is computationally significantly more demanding.

6. Quadratic assignments. The quadratic assignment problem, or QAP for short, is a generalization of the (linear) assignment problem, briefly described in the introduction. It consists of minimizing a quadratic function over the set of permutation matrices $\{X_\phi : \phi \in \Pi\}$. The general form of QAP, for given symmetric $n^2 \times n^2$ matrix Q is as follows

$$z_{QAP} := \min\{x_\phi^T Q x_\phi : \quad x_\phi = \text{vec}(X_\phi), \quad \phi \in \Pi\}.$$

In applications, Q is often of the form $Q = B \otimes A$, where A and B are symmetric $n \times n$ matrices. In this case the objective function has a representation in terms of $n \times n$ matrices, see (5.8)

$$x_\phi^T (B \otimes A) x_\phi = \langle X_\phi, AX_\phi B \rangle.$$

We refer to the recent monograph [12] by Burkard et al for further details on assignment problems.

To get a handle on QAP, we consider the matrix lifting

$$\mathcal{M}_{QAP} := \text{conv}\{x_\phi x_\phi^T : x_\phi = \text{vec}(X_\phi), \phi \in \Pi\}.$$

We will now see that \mathcal{M}_{QAP} has a 'simple' description by linear equations intersected with the cone C^* . Matrices $Y \in \mathcal{M}_{QAP}$ are of order $n^2 \times n^2$. It will be useful to consider the following partitioning of Y into $n \times n$ block matrices $Y^{i,j}$,

$$Y = \begin{pmatrix} Y^{1,1} & \dots & Y^{1,n} \\ \vdots & \ddots & \vdots \\ Y^{n,1} & \dots & Y^{n,n} \end{pmatrix}.$$

Let $X = (x_1, \dots, x_n)$ be a permutation matrix (with columns x_i). Then X can be characterized by $X \geq 0$, $X^T X = X X^T = I$. These quadratic constraints translate into linear constraints on Y :

$$X X^T = \sum_i x_i x_i^T = \sum_i Y^{i,i} = I.$$

Similarly, $(X^T X)_{ij} = x_i^T x_j = \text{tr}(x_i x_j^T) = \text{tr}(Y^{i,j}) = \delta_{ij}$. Finally, we have $(\sum_{ij} x_{ij})^2 = n^2$ for any permutation matrix X . We get the following set of constraints for Y :

$$\sum_i Y^{i,i} = I, \quad \text{tr}(Y^{i,j}) = \delta_{ij}, \quad \langle J, Y \rangle = n^2. \quad (6.1)$$

Povh and Rendl [55] show the following characterization of \mathcal{M}_{QAP} , which can be viewed as a lifted version of Birkhoff's theorem 1.1.

THEOREM 6.1. $\mathcal{M}_{QAP} = \{Y : Y \in C^*, Y \text{ satisfies (6.1)}\}$.

It is not hard to verify that the above result would be wrong without the seemingly redundant equation $\langle J, Y \rangle = n^2$. We can therefore formulate the quadratic problem QAP as a (linear but intractable) copositive program

$$z_{QAP} = \min\{\langle Q, Y \rangle : Y \in C^*, Y \text{ satisfies (6.1)}\}.$$

In [55] some semidefinite relaxations based on this model are investigated and compared to previously published SDP relaxations of QAP.

We have now seen several instances of combinatorial optimization problems, where CP relaxations in fact gave the exact value. This raises the

question whether there is some general principle behind this observation. Burer [10] gives a rather general answer and shows that an appropriate reformulation of quadratic programs is equivalent to a linear copositive program.

THEOREM 6.2. [10] *Let c and a_j be vectors from \mathbb{R}^n , $b \in \mathbb{R}^k$, $Q \in \mathcal{S}_n$ and $I \subseteq \{1, \dots, n\}$. The optimal values of the following two problems are equal*

$$\min\{x^T Qx + c^T x : a_j^T x = b_j, x \geq 0, x_i \in \{0, 1\} i \in I\},$$

$$\min\{\langle Q, X \rangle + c^T x : a_j^T x = b_j, a_j^T X a_j = b_j^2,$$

$$X_{ii} = x_i \quad \forall i \in I, \quad \left(\begin{array}{cc} 1 & x^T \\ x & X \end{array} \right) \in \mathcal{C}^* \}.$$

7. Optimal mixing rate of Markov chains. In the previous sections we focused on various ways to get matrix liftings of NP-hard optimization problems. We conclude now with an SDP model which optimizes the mixing rate of a finite Markov chain. Let G be a connected graph. We consider random walks on the vertex set $V(G)$. Suppose we can either stay at $i \in V(G)$ or move to j , provided $ij \in E(G)$. Suppose further that the transition probabilities p_{ij} are symmetric, $p_{ij} = p_{ji} \forall ij$. The resulting Markov chain is now described by the transition matrix P satisfying

$$P \in \mathcal{P}_G := \{P : P = P^T, P \in \Omega, p_{ij} = 0 \text{ if } ij \notin E(G)\}.$$

Finally, we assume that P is primitive (aperiodic and irreducible). This means there exists some k such that $P^k > 0$. Let us first recall the Perron-Frobenius theorem for primitive matrices.

THEOREM 7.1. *Let P be a nonnegative primitive square matrix. Then the spectral radius is a simple eigenvalue of P with eigenvector $x > 0$.*

We denote by $\pi(t) \in \mathbb{R}^n$ the probability distribution on V at time t . The definition of the transition probabilities in P imply that $\pi(t+1) = P^T \pi(t)$. Symmetry of P therefore shows that $\pi(t)$ is determined from the initial distribution $\pi(0)$ through $\pi(t) = P^t \pi(0)$. We denote the eigenvalues of P by

$$1 = \lambda_1(P) > \lambda_2(P) \geq \dots \geq \lambda_n(P) > -1.$$

The Perron-Frobenius theorem tells us that $\frac{1}{n}e$ is eigenvector to the eigenvalue 1, which is also the spectral radius of P . Therefore

$$\lim_{t \rightarrow \infty} \pi(t) = \frac{1}{n}e.$$

What can be said about the speed of convergence? There are several ways to measure the distance of $\pi(t)$ from the equilibrium distribution $\pi(\infty) = \pi = \frac{1}{n}e$. One such measure is the maximum relative error at time t

$$r(t) := \max_{ij} \frac{|(P^t)_{ij} - \pi_j|}{\pi_j},$$

see for instance [3]. Let

$$\mu_P := \max\{\lambda_2(P), -\lambda_n(P)\} = \max\{|\lambda_i(P)| : i > 1\}$$

denote the second largest eigenvalue of P in modulus (SLEM). It is well known that μ_P is closely related to how fast $\pi(t)$ converges to the equilibrium distribution $\frac{1}{n}e$.

THEOREM 7.2. [3] *Let P be a symmetric irreducible transition matrix. Then*

$$r(t) \leq n(\mu_P)^t.$$

Moreover $r(t) \geq (\mu_P)^t$ if t is even.

Given the graph G , we can ask the question to select the transition probabilities $p_{ij} > 0$ for $ij \in E(G)$ in such a way that the mixing rate of the Markov chain is as fast as possible. In view of the bounds from the previous theorem, it makes sense to consider the following optimization problem (in the matrix variable P), see Boyd et. al. [9]

$$\min\{\mu_P : P \in \mathcal{P}_G\}.$$

They show that μ_P is in fact a convex function. It follows from the Perron-Frobenius theorem and the spectral decomposition theorem for symmetric matrices that μ_P is either the smallest or the largest eigenvalue of $P - \frac{1}{n}J$ in absolute value. Hence we can determine μ_P as the solution of the following SDP, see [9]

$$\min\{s : sI \succeq P - \frac{1}{n}J \succeq -sI, P \in \mathcal{P}_G\}. \quad (7.1)$$

The variables are s and the matrix P . In [9], it is shown that an optimal choice of P may significantly increase the mixing rate of the resulting chain. Some further extensions of this idea are discussed in [64].

8. Computational progress. Up to now we have mostly concentrated on the modeling power of SDP and CP. From a practical point of view, it is also important to investigate the algorithmic possibilities to actually solve the relaxations. Solving copositive programs is at least as hard as general integer programming, hence we only consider solving SDP, which are tractable. Before we describe various algorithms to solve SDP, we recall the basic assumptions from Theorem 2.1,

$$\exists X_0 \succ 0, y_0 \text{ such that } A(X) = b, C - A^T(y_0) \succ 0. \quad (8.1)$$

In this case (X, y, Z) is optimal for the primal-dual pair

$$\min\{\langle C, X \rangle : A(X) = b, X \succeq 0\} = \max\{b^T y : C - A^T(y) = Z \succeq 0\}$$

if and only if

$$A(X) = b, A^T(y) + Z = C, X \succeq 0, Z \succeq 0, \langle X, Z \rangle = 0. \quad (8.2)$$

We are now going to briefly describe several classes of algorithms to solve SDP and point out their strengths and limitations.

8.1. Interior-point methods. There exist several quite in-depth descriptions of primal-dual interior-point path-following methods for SDP. We refer to [67, 63, 14] and the SDP handbook [70]. The website ³ maintains a collection of software, various data sets and provides benchmark comparisons of several competing software packages to solve SDP.

We therefore explain here only the basic ideas. First, it follows from $X \succeq 0, Z \succeq 0, \langle X, Z \rangle = 0$ that in fact $XZ = 0$. Interior-point (path-following) methods can be viewed as a sequence of problems parametrized by $\mu \geq 0$. Consider the set P_μ , defined as follows:

$$P_\mu := \{(X, y, Z) : A(X) = b, Z + A^T(y) = C, X \succeq 0, Z \succeq 0, ZX = \mu I\}.$$

Clearly, $P_0 \neq \emptyset$, if (8.1) holds. It can in fact be shown that P_μ defines a unique point (X_μ, y_μ, Z_μ) for any $\mu > 0$ if and only if (8.1) holds. See for instance Theorem 10.2.1 in [70]. In this case the set $\{(X_\mu, y_\mu, Z_\mu) : \mu \geq 0\}$ defines a smooth curve, parametrized by $\mu > 0$. Following this path until $\mu \approx 0$ clearly leads to an optimal solution of SDP. This is the basic idea underlying interior-point methods. There is quite a variety of different approaches to achieve this goal. Todd [65] gives a detailed summary of popular variants to solve SDP by path-following methods.

The crucial step in all these variants consists of the following. Given a current iterate (X_k, y_k, Z_k) with $X_k \succ 0$ and $Z_k \succ 0$ and a target path parameter $\mu_k > 0$, we use the Newton method to determine a search direction $(\Delta X, \Delta y, \Delta Z)$ towards (the point) P_{μ_k} . If there are m equations in the primal problem, so $b \in \mathbb{R}^m$, this amounts to setting up and solving a (dense) linear system of order m to determine Δy . To set up this system, and to recover ΔX and ΔZ , some additional computational effort is necessary, involving matrix operations (multiplication, inversion) with matrices of order n . Having the search direction, one needs to test whether the full Newton step is feasible ($X_k + \Delta X \succ 0$ and $Z_k + \Delta Z \succ 0$). If not, some sort of backtracking strategy is used to find a smaller steplength leading to a new iterate in the interior of \mathcal{S}^+ . Then a new (smaller) target value for μ is selected and the process is iterated until $\mu \approx 0$ and the current iterates are primal and dual feasible.

³<http://plato.asu.edu/bench.html>.

TABLE 1

Interior-point computation times to solve (3.7) with relative accuracy 10^{-6} . Here $m = n$.

n	time (secs.)
1000	12
2000	102
3000	340
4000	782
5000	1570

TABLE 2

Interior-point computation times to solve (4.2) with relative accuracy 10^{-6} , $m = \frac{1}{2}\binom{n}{2}$ and $m = 5n$.

n	$m = \frac{1}{2}\binom{n}{2}$	time (secs.)	n	$m = 5n$	time (secs.)
100	2488	12	500	2500	14
150	5542	125	1000	5000	120
200	9912	600	1500	7500	410

The convergence analysis shows that under suitable parameter settings it takes $O(\sqrt{n})$ Newton iterations to reach a solution with the required accuracy. Typically, the number of such iterations is not too large, often only a few dozen, but both the memory requirements (a dense $m \times m$ matrix has to be handled) and the computation times grow rapidly with n and m . To give some impression, we provide in Table 1 some sample timings to solve the basic relaxation for Max-Cut, see (3.7). It has $m = n$ rather simple constraints $x_{ii} = 1$. We also consider computing the theta number $\vartheta(G)$ (4.2), see Table 2. Here the computational effort is also influenced by the cardinality $|E(G)|$. We consider dense graphs ($m = \frac{1}{2}\binom{n}{2}$) and sparse graphs ($m = 5n$). In the first case, the number n of vertices can not be much larger than about 200, in the second case we can go to much larger graphs. Looking at these timings, it is quite clear that interior-point methods will become impractical once $n \approx 3000$ or $m \approx 5000$.

There have been attempts to overcome working explicitly with the dense system matrix of order m . Toh [66] for instance reports quite encouraging results for larger problems by iteratively solving the linear system for the search direction. A principal drawback of this approach lies in the fact that the system matrix gets ill-conditioned, as one gets close to the optimum. This implies that high accuracy is not easily reachable. We also mention the approach from Kocvara and Stingl [38], which uses a modified ‘barrier function’ and also handles large-scale problems. Another line of research to overcome some of these limitations consists in exploiting sparsity in the data. We refer to [20, 50] for some first fundamental steps in this direction.

8.2. Projection methods. To overcome some of the computational bottlenecks of interior-point methods, we can exploit the fact that the projection of an arbitrary symmetric matrix M to the cone of semidefinite matrices can be obtained through a spectral decomposition of M . More precisely, let $M = \sum_i \lambda_i u_i u_i^T$ with pairwise orthogonal eigenvectors u_i . Then

$$\operatorname{argmin}\{\|M - X\| : X \succeq 0\} = \sum_{i:\lambda_i > 0} \lambda_i u_i u_i^T =: M^+, \quad (8.3)$$

see for instance [24].

A rather natural use of projection was recently proposed in [34] and can be explained as follows. We recall the optimality conditions (8.2) and observe that $\langle X, Z \rangle = 0$ can be replaced by the linear equation $b^T y - \langle C, X \rangle = 0$. Hence we can group the optimality conditions into the affine linear constraints

$$(L_P) \quad A(X) = b, \quad (L_D) \quad A^T(y) + Z = C, \quad (L_C) \quad \langle C, X \rangle - b^T y = 0,$$

and the SDP conditions $X \succeq 0$, $Z \succeq 0$. The projection onto SDP is given by (8.3). Projecting onto an affine space is also quite easy. Linear algebra tells us that the projection $\Pi_P(X)$ of a symmetric matrix X onto (L_P) is given by

$$\Pi_P(X) := X - A^T(AA^T)^{-1}(A(X) - b),$$

and similarly, Z has the projection

$$\Pi_D(Z) := C + A^T(AA^T)^{-1}A(Z - C)$$

onto L_D . Thus $\Pi_D(Z) = C + A^T(y)$ with $y = (AA^T)^{-1}A(Z - C)$. Finally, the projection onto the hyperplane L_C is trivial. Thus one can use alternate projections to solve SDP. Take a starting point (X, y, Z) , and project it onto the affine constraints. This involves solving two linear equations with system matrix AA^T , which remains unchanged throughout. Then project the result onto the SDP cone and iterate. This requires the spectral decomposition of both X and Z .

This simple iterative scheme is known to converge slowly. In [34] some acceleration strategies are discussed and computational results with $m \approx 100,000$ are reported.

Another solution approach for SDP using only SDP projection and solving a linear equation with system matrix AA^T is proposed by Povh et al [56] and Malick et al [46]. The approach from [34] can be viewed as maintaining $A(X) = b$, $Z + A^T(y) = C$ and the zero duality gap condition $b^T y = \langle C, X \rangle$ and trying to get X and Z into S^+ . In contrast, the approach from [56, 46] maintains $X \succeq 0$, $Z \succeq 0$, $ZX = 0$ and tries to reach feasibility with respect to the linear equations. The starting point of this approach

consists of looking at the augmented Lagrangian formulation of the dual SDP. Let

$$f_{\sigma,X}(y, Z) := b^T y - \langle X, A^T(y) + Z - C \rangle - \frac{\sigma}{2} \|A^T(y) + Z - C\|^2 \quad (8.4)$$

and consider

$$\max_{y, Z \succeq 0} f_{\sigma,X}(y, Z).$$

Having (approximate) maximizers y, Z (for σ and X held constant), the augmented Lagrangian method, see [5], asks to update X by

$$X \leftarrow X + \sigma(Z + A^T(y) - C) \quad (8.5)$$

and iterate until dual feasibility is reached. The special structure of the subproblem given by (8.4) allows us to interpret the update (8.5) differently. After introducing the Lagrangian $L = f_{\sigma,X}(y, Z) + \langle V, Z \rangle$ with respect to the constraint $Z \succeq 0$, we get the following optimality conditions for maximizing (8.4)

$$\nabla_y L = b - A(X) - \sigma A(A^T(y) + Z - C) = 0,$$

$$\nabla_Z L = V - X - \sigma(A^T(y) + Z - C) = 0, \quad V \succeq 0, \quad Z \succeq 0, \quad \langle V, Z \rangle = 0.$$

The condition $\nabla_Z L = 0$ suggests to set $X = V$, see (8.5). We note that L could also be written as

$$L = b^T y - \frac{\sigma}{2} \|Z - (C - A^T(y) - \frac{1}{\sigma} X)\|^2 + \frac{1}{2\sigma} \|X\|^2.$$

Therefore, at the optimum, Z must also be the projection of $W := C - A^T(y) - \frac{1}{\sigma} X$ onto \mathcal{S}^+ , $Z = W^+$. Thus $\nabla_Z L = 0$ becomes

$$V = \sigma(Z + \frac{1}{\sigma} X + A^T(y) - C) = \sigma(W^+ - W) = -\sigma W^-,$$

where W^- is the projection of W onto $-(\mathcal{S}^+)$. This leads to the boundary point method from [56]. Given X, Z , solve the $\nabla_y L = 0$ for y :

$$AA^T(y) = \frac{1}{\sigma}(b - A(X) - A(Z - C)).$$

Then compute the spectral decomposition of $W = C - A^T(y) - \frac{1}{\sigma} X$ and get a new iterate $X = -\sigma W^-, Z = W^+$ and iterate.

The computational effort of one iteration is essentially solving the linear system with matrix AA^T and computing the factorization of W . Further details, like convergence analysis and parameter updates are described in [56, 46]. To show the potential of this approach, we compute $\vartheta(G)$ for

TABLE 3
Boundary-point computation times to solve (4.2) with relative accuracy 10^{-6} .

n	$m = \frac{1}{4}n^2$	time (secs.)
400	40000	40
600	90000	100
800	160000	235
1000	250000	530
1200	360000	1140

larger dense graphs with $m = \frac{1}{4}n^2$, see Table 3. It is clear from these results that projection methods extend the computational possibilities for SDP. Zhao et al [71] recently generalized this approach to include second order information in the updates. At higher computational cost they get more accurate solutions.

The computational limitations of projection methods are determined on one hand by the need to compute a spectral decomposition of a symmetric matrix, which limits the matrix dimension similar to interior-point methods. On the other hand, the system matrix AA^T does not change during the algorithm, and any sparsity properties of the input can therefore be fully exploited. In case of computing the ϑ function for instance, it turns out that AA^T is in fact diagonal. This is one explanation why instances with m beyond 100,000 are easily manageable by projection methods.

8.3. Further solution approaches for SDP. To avoid limits on the size of primal matrices (given through the spectral decomposition or the linear algebra of interior-point methods), one can also use the encoding of semidefiniteness through the condition $\lambda_{\min}(X) \geq 0$. The computation of $\lambda_{\min}(X)$, for symmetric X , can be done iteratively for quite large matrices. The only requirement is that $v = Xu$ can be evaluated. In particular, X need not be stored explicitly. The **spectral bundle method** exploits this fact, and applies to SDP, where (primal) feasibility implies constant trace. It was introduced in [32] and we refer to this paper for all further details. It turns out that SDP with rather large matrices ($n \approx 10,000$) can be handled by this method, but the convergence properties are much weaker than in the case of interior-point methods. Helmberg [30] describes computational results on a variety of large scale combinatorial optimization problems.

Another idea to get rid of the semidefiniteness condition is to use the factorization $X = RR^T$, and work with R using algorithms from nonlinear optimization. Burer and Monteiro [11] investigate this approach and present some encouraging computational results for some specially structured SDP like the Max-Cut relaxation (3.7). The drawback here is that by going from X to the factor R , one loses convexity.

Finally, SDP based relaxations have been used successfully to get exact solutions. Exact solutions of the Max-Cut problem are reported in [58]

for instances having several hundred vertices, see also the thesis [69]. A combination of polyhedral and SDP relaxations for the bisection problem is studied in [1]. Exact solutions of rather large sparse instances ($n \approx 1000$) are obtained for the first time. Finally, exact solutions for Max- k -Cut are given in [21].

Algorithms for linear optimization have reached a high level of sophistication, making them easily accessible even for non-experts. In contrast, most algorithms for SDP require a basic understanding of semidefinite optimization. While small problems $n \approx 100$ can be solved routinely, there is no standard way to solve medium sized SDP in a routine way. Finally, it is a challenging open problem to find efficient ways of optimizing over $\mathcal{S}^+ \cap \mathcal{N}$.

Acknowledgements. Many thanks go to Nathan Krislock and an anonymous referee for giving numerous suggestions for improvement, and for pointing out minor inconsistencies.

REFERENCES

- [1] M. ARMBRUSTER, M. FÜGENSCHUH, C. HELMBERG, AND A. MARTIN. A comparative study of linear and semidefinite branch-and-cut methods for solving the minimum graph bisection problem. In *Integer programming and combinatorial Optimization (IPCO 2008)*, pages 112–124, 2008.
- [2] S. ARORA, E. CHLAMTAC, AND M. CHARIKAR. New approximation guarantee for chromatic number. In *Proceedings of the 38th STOC, Seattle, USA*, pages 215–224, 2006.
- [3] E. BEHREND. *Introduction to Markov chains with special emphasis on rapid mixing*. Vieweg Advanced Lectures in Mathematics, 2000.
- [4] A. BERMAN AND N. SHAKED-MONDERER. *Completely positive matrices*. World Scientific Publishing, 2003.
- [5] D.P. BERTSEKAS. *Constrained optimization and Lagrange multipliers*. Academic Press, 1982.
- [6] A. BLUM AND D. KARGER. An $\tilde{O}(n^{3/14})$ -coloring algorithm for 3-colorable graphs. *Information processing letters*, **61**(1):249–53, 1997.
- [7] A. BLUM, G. KONJEVOD, R. RAVI, AND S. VEMPALA. Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems. *Theoretical Computer Science*, **235**(1):25–42, 2000.
- [8] I.M. BOMZE, M. DÜR, E. DE KLERK, C. ROOS, A.J. QUIST, AND T. TERLAKY. On copositive programming and standard quadratic optimization problems. *Journal of Global Optimization*, **18**(4):301–320, 2000.
- [9] S. BOYD, P. DIACONIS, AND L. XIAO. Fastest mixing Markov chain on a graph. *SIAM Review*, **46**(4):667–689, 2004.
- [10] S. BURER. On the copositive representation of binary and continuous nonconvex quadratic programs. *Mathematical Programming (A)*, **120**:479–495, 2009.
- [11] S. BURER AND R.D.C. MONTEIRO. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (B)*, **95**:329–357, 2003.
- [12] R.E. BURKARD, M. DELL'AMICO, AND S. MARTELLO. *Assignment problems*. SIAM, 2009.
- [13] E. CHLAMTAC. *Non-local Analysis of SDP based approximation algorithms*. PhD thesis, Princeton University, USA, 2009.

- [14] E. DE KLERK. *Aspects of semidefinite programming: interior point algorithms and selected applications*. Kluwer Academic Publishers, 2002.
- [15] E. DE KLERK AND D.V. PASECHNIK. Approximation of the stability number of a graph via copositive programming. *SIAM Journal on Optimization*, **12**:875–892, 2002.
- [16] E. DE KLERK, D.V. PASECHNIK, AND J.P. WARNERS. On approximate graph colouring and Max- k -Cut algorithms based on the ϑ -function. *Journal of Combinatorial Optimization*, **8**(3):267–294, 2004.
- [17] R.J. DUFFIN. Infinite programs. *Ann. Math. Stud.*, **38**:157–170, 1956.
- [18] I. DUKANOVIC AND F. RENDL. Copositive programming motivated bounds on the stability and chromatic numbers. *Mathematical Programming*, **121**:249–268, 2010.
- [19] A. FRIEZE AND M. JERRUM. Improved approximation algorithms for Max k -Cut and Max Bisection. *Algorithmica*, **18**(1):67–81, 1997.
- [20] M. FUKUDA, M. KOJIMA, K. MUROTA, AND K. NAKATA. Exploiting sparsity in semidefinite programming via matrix completion. I: General framework. *SIAM Journal on Optimization*, **11**(3):647–674, 2000.
- [21] B. GHADDAR, M. ANJOS, AND F. LIERS. A branch-and-cut algorithm based on semidefinite programming for the minimum k -partition problem. *Annals of Operations Research*, 2008.
- [22] M.X. GOEMANS. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, **79**:143–162, 1997.
- [23] M.X. GOEMANS AND D.P. WILLIAMSON. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, **42**:1115–1145, 1995.
- [24] G. GOLUB AND C.F. VAN LOAN. *Matrix computations. 3rd ed.* Baltimore, MD: The Johns Hopkins Univ. Press., 1996.
- [25] M. GRÖTSCHEL, L. LOVÁSZ, AND A. SCHRIJVER. *Geometric algorithms and combinatorial optimization*. Springer, Berlin, 1988.
- [26] N. GVOZDENOVIĆ. *Approximating the stability number and the chromatic number of a graph via semidefinite programming*. PhD thesis, University of Amsterdam, 2008.
- [27] N. GVOZDENOVIĆ AND M. LAURENT. Computing semidefinite programming lower bounds for the (fractional) chromatic number via block-diagonalization. *SIAM Journal on Optimization*, **19**(2):592–615, 2008.
- [28] N. GVOZDENOVIĆ AND M. LAURENT. The operator Ψ for the chromatic number of a graph. *SIAM Journal on Optimization*, **19**(2):572–591, 2008.
- [29] C. HELMBERG. Semidefinite programming. *European Journal of Operational Research*, **137**:461–482, 2002.
- [30] C. HELMBERG. Numerical validation of SBmethod. *Mathematical Programming*, **95**:381–406, 2003.
- [31] C. HELMBERG, B. MOHAR, S. POLJAK, AND F. RENDL. A spectral approach to bandwidth and separator problems in graphs. *Linear and Multilinear Algebra*, **39**:73–90, 1995.
- [32] C. HELMBERG AND F. RENDL. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, **10**:673–696, 2000.
- [33] J.-B. HIRIART-URRUTY AND A. SEEGER. A variational approach to copositive matrices. *SIAM Review*, **52**(4):593–629, 2010.
- [34] F. JARRE AND F. RENDL. An augmented primal-dual method for linear conic problems. *SIAM Journal on Optimization*, **20**:808–823, 2008.
- [35] D. KARGER, R. MOTWANI, AND M. SUDAN. Approximate graph colouring by semidefinite programming. *Journal of the ACM*, **45**:246–265, 1998.
- [36] S.E. KARISCH AND F. RENDL. Semidefinite Programming and Graph Equipartition. *Fields Institute Communications*, **18**:77–95, 1998.
- [37] S. KHANNA, N. LINIAL, AND S. SAFRA. On the hardness of approximating the chromatic number. *Combinatorica*, **20**:393–415, 2000.

- [38] M. KOVARA AND M. STINGL. On the solution of large-scale SDP problems by the modified barrier method using iterative solvers. *Mathematical Programming*, **95**:413–444, 2007.
- [39] M. LAURENT AND F. RENDL. Semidefinite programming and integer programming. In K. Aardal, G.L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, pages 393–514. Elsevier, 2005.
- [40] E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOY KAN, AND D.B. SHMOYS (eds.). *The traveling salesman problem, a guided tour of combinatorial optimization*. Wiley, Chichester, 1985.
- [41] A. LISSEK AND F. RENDL. Graph partitioning using Linear and Semidefinite Programming. *Mathematical Programming (B)*, **95**:91–101, 2002.
- [42] L. LOVÁSZ. On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory*, **25**:1–7, 1979.
- [43] L. LOVÁSZ. Semidefinite programs and combinatorial optimization. In B. A. Reed and C.L. Sales, editors, *Recent advances in algorithms and combinatorics*, pages 137–194. CMS books in Mathematics, Springer, 2003.
- [44] L. LOVÁSZ AND A. SCHRIJVER. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, **1**:166–190, 1991.
- [45] D. JOHNSON M. GAREY, R. GRAHAM AND D.E. KNUTH. Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics*, **34**:477–495, 1978.
- [46] J. MALICK, J. POVH, F. RENDL, AND A. WIEGELE. Regularization methods for semidefinite programming. *SIAM Journal on Optimization*, **20**:336–356, 2009.
- [47] R.J. MCELIECE, E.R. RODEMICH, AND H.C. RUMSEY JR. The Lovász bound and some generalizations. *Journal of Combinatorics and System Sciences*, **3**:134–152, 1978.
- [48] T.S. MOTZKIN AND E.G. STRAUS. Maxima for graphs and a new proof of a theorem of Turán. *Canadian Journal of Mathematics*, **17**:533–540, 1965.
- [49] K.G. MURTY AND S.N. KABADI. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, **39**:117–129, 1987.
- [50] K. NAKATA, K. FUJISAWA, M. FUKUDA, K. KOJIMA, AND K. MUROTA. Exploiting sparsity in semidefinite programming via matrix completion. II: Implementation and numerical results. *Mathematical Programming*, **95**:303–327, 2003.
- [51] Y. NESTEROV. Quality of semidefinite relaxation for nonconvex quadratic optimization. Technical report, CORE, 1997.
- [52] C.H. PAPADIMITRIOU. The NP-completeness of the bandwidth minimization problem. *Computing*, **16**:263–270, 1976.
- [53] J. POVH. *Applications of semidefinite and copositive programming in combinatorial optimization*. PhD thesis, University of Ljubljana, Slovenia, 2006.
- [54] J. POVH AND F. RENDL. A copositive programming approach to graph partitioning. *SIAM Journal on Optimization*, **18**(1):223–241, 2007.
- [55] J. POVH AND F. RENDL. Copositive and semidefinite relaxations of the quadratic assignment problem. *Discrete Optimization*, **6**(3):231–241, 2009.
- [56] J. POVH, F. RENDL, AND A. WIEGELE. A boundary point method to solve semidefinite programs. *Computing*, **78**:277–286, 2006.
- [57] F. RENDL. Semidefinite relaxations for integer programming. In M. Jünger Th.M. Liebling D. Naddef G.L. Nemhauser W.R. Pulleyblank G. Reinelt G. Rinaldi and L.A. Wolsey, editors, *50 years of integer programming 1958–2008*, pages 687–726. Springer, 2009.
- [58] F. RENDL, G. RINALDI, AND A. WIEGELE. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, **212**:307–335, 2010.
- [59] A. SCHRIJVER. A comparison of the Delsarte and Lovász bounds. *IEEE Transactions on Information Theory*, **IT-25**:425–429, 1979.
- [60] A. SCHRIJVER. *Combinatorial optimization. Polyhedra and efficiency*. Vol. B, volume **24** of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2003.

- [61] H.D. SHERALI AND W.P. ADAMS. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, **3**(3):411–430, 1990.
- [62] H.D. SHERALI AND W.P. ADAMS. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Appl. Math.*, **52**(1):83–106, 1994.
- [63] J. STURM. Theory and algorithms of semidefinite programming. In T. Terlaki H. Frenk, K.Roos and S. Zhang, editors, *High performance optimization*, pages 1–194. Springer Series on Applied Optimization, 2000.
- [64] J. SUN, S. BOYD, L. XIAO, AND P. DIACONIS. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, **48**(4):681–699, 2006.
- [65] M.J. TODD. A study of search directions in primal-dual interior-point methods for semidefinite programming. *Optimization Methods and Software*, **11**:1–46, 1999.
- [66] K.-C. TOH. Solving large scale semidefinite programs via an iterative solver on the augmented systems. *SIAM Journal on Optimization*, **14**:670 – 698, 2003.
- [67] L. VANDENBERGHE AND S. BOYD. Semidefinite programming. *SIAM Review*, **38**:49–95, 1996.
- [68] A. WIDGERSON. Improving the performance guarantee for approximate graph colouring. *Journal of the ACM*, **30**:729– 735, 1983.
- [69] A. WIEGELE. *Nonlinear optimization techniques applied to combinatorial optimization problems*. PhD thesis, Alpen-Adria-Universität Klagenfurt, Austria, 2006.
- [70] H. WOLKOWICZ, R. SAIGAL, AND L. VANDENBERGHE (eds.). *Handbook of semidefinite programming*. Kluwer, 2000.
- [71] X. ZHAO, D. SUN, AND K. TOH. A Newton CG augmented Lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, **20**:1737–1765, 2010.

A POLYTOPE FOR A PRODUCT OF REAL LINEAR FUNCTIONS IN 0/1 VARIABLES

OKTAY GÜNLÜK*, JON LEE†, AND JANNA LEUNG‡

Abstract. In the context of integer programming, we develop a polyhedral method for linearizing a product of a pair of real linear functions in 0/1 variables. As an example, by writing a pair of integer variables in binary expansion, we have a technique for linearizing their product. We give a complete linear description for the resulting polytope, and we provide an efficient algorithm for the separation problem. Along the way to establishing the complete description, we also give a complete description for an extended-variable formulation, and we point out a generalization.

Key words. Integer nonlinear programming, polytope, product.

AMS(MOS) subject classifications. 52B11, 90C10, 90C57, 90C30.

1. Introduction. We assume familiarity with polyhedral methods of linear integer programming (see [NW88], for example). There is a well-known method of linear integer programming for modeling the product (i.e., logical AND) of a *pair* of binary variables. Specifically, the 0/1 solutions of $y = x^1 x^2$ are, precisely, the extreme points of the polytope in \mathbb{R}^3 that is the solution set of

$$y \geq 0 ; \tag{1.1}$$

$$y \geq x^1 + x^2 - 1 ; \tag{1.2}$$

$$y \leq x^1 ; \tag{1.3}$$

$$y \leq x^2 . \tag{1.4}$$

There has been extensive interest in the development of linear integer programming methods for handling and/or exploiting products of *many* 0/1 variables. In particular, quite a lot is known about the facets of the convex hull of the 0/1 solutions to: $y^{ij} = x^i x^j$, $1 \leq i < j \leq n$ (the *boolean quadric polytope*). See [Pad89]. This polytope is related to other well-studied structures such as the cut polytope [DS90] and the correlation polytope [Pit91]. Also see [BB98], [DL97] and references therein. Of course, we know less and less about the totality of the facets of the polytope as n increases, because optimization over these 0/1 solutions is NP-hard.

Rather than considering pairwise products of many 0/1 variables, we consider a single product of a pair of real linear functions in 0/1 variables. For $l = 1, 2$, let k_l be a positive integer, let $K_l = \{1, 2, \dots, k_l\}$, let \mathbf{a}^l be

*IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A.
(gunluk@us.ibm.com).

†Department of Industrial and Operations Engineering, University of Michigan,
Ann Arbor, MI 48109, U.S.A. (jonxlee@umich.edu).

‡Chinese University of Hong Kong (janny@se.cuhk.edu.hk).

a k_l -vector of positive real numbers a_i^l , and let \mathbf{x}^l be a k_l -vector of binary variables x_i^l . Note that if we had any $a_i^l = 0$, then we could just delete such x_i^l , and if we had any $a_i^l < 0$, then we could just complement such x_i^l and apply our methods to the nonlinear part of the resulting product. So, hereafter, we assume without loss of generality that a_i^l are all positive.

Now, we let $P(\mathbf{a}^1, \mathbf{a}^2)$ be the convex hull of the solutions in $\mathbb{R}^{k_1+k_2+1}$ of

$$y = \left(\sum_{i \in K_1} a_i^1 x_i^1 \right) \left(\sum_{j \in K_2} a_j^2 x_j^2 \right) = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^1 x_j^2 ; \quad (1.5)$$

$$x_i^l \in \{0, 1\}, \text{ for } i \in K_l, l = 1, 2 . \quad (1.6)$$

We note that $P((1), (1))$ is just the solution set of (1.1–1.4). Our goal is to investigate the polytope $P(\mathbf{a}^1, \mathbf{a}^2)$ generally.

In Section 2, we describe an application to modeling a product of a pair of nonnegative integer variables using binary expansion. In Section 3, we describe a linear integer formulation of $P(\mathbf{a}^1, \mathbf{a}^2)$. In Section 4, we investigate which of our inequalities are facet describing. In Section 5, we determine a complete polyhedral characterization of $P(\mathbf{a}^1, \mathbf{a}^2)$. In establishing this characterization, we also find an inequality characterization of a natural extended-variable formulation. In Section 6, we demonstrate how to solve the separation problem for the facet describing inequalities of $P(\mathbf{a}^1, \mathbf{a}^2)$. In Section 7, we investigate some topological properties of *real* points in the $P(\mathbf{a}^1, \mathbf{a}^2)$ that satisfy the product equation (1.5). In Section 8, we briefly describe a generalization of our results.

Our results were first reported in complete form in [CGLL03], which in turn was developed from [CLL99]. Other related work, much of which was undertaken or announced later, includes [BB98, AFLS01, HHPR02, JKKW05, Kub05, CKM06, FFL05, Lee07].

2. Products of nonnegative integer variables. Let x^1 and x^2 be a pair of nonnegative integer variables. We can look directly at the convex hull of the integer solutions of $y = x^1 x^2$. This convex set, in \mathbb{R}^3 , contains integer points that do not correspond to solutions of $y = x^1 x^2$. For example, $x^1 = x^2 = y = 0$ is in this set, as is $x^1 = x^2 = 2, y = 4$, but the average of these two points is not a solution of $y = x^1 x^2$. More concretely, the convex hull of the integer solutions to:

$$\begin{aligned} y &= x^1 x^2 ; \\ 0 &\leq x^1 \leq 2 ; \\ 0 &\leq x^2 \leq 2 \end{aligned}$$

is precisely the solution set of

$$\begin{aligned} y &\geq 0 ; \\ y &\geq 2x^1 + 2x^2 - 4 ; \\ y &\leq 2x^1 ; \\ y &\leq 2x^2 . \end{aligned}$$

If we use these latter linear inequalities to model the product y , and then seek to maximize y subject to these constraints and a side constraint $x^1 + x^2 \leq 2$, we find the optimal solution $x^1 = 1$, $x^2 = 1$, $y = 2$, which does not satisfy $y = x^1 x^2$. Therefore, this naïve approach is inadequate in the context of linear integer programming.

We adopt an approach that avoids the specific problem above where an integer infeasible point is caught in the convex hull of integer feasible solutions. Specifically, we assume that, for practical purposes, x^1 and x^2 can be bounded above. So we can write the x^l in binary expansion: $x^l = \sum_{i \in K_l} 2^{i-1} x_i^l$, for $l = 1, 2$. That is, we let $a_i^l = 2^{i-1}$. The only integer points in $P(\mathbf{a}^1, \mathbf{a}^2)$ are the solutions of (1.5–1.6). Therefore, we avoid the problem that we encountered when we did not use the binary expansions of x^1 and x^2 .

3. Linear integer formulation. Obviously, for $l = 1, 2$, the *simple bound inequalities* are valid for $P(\mathbf{a}^1, \mathbf{a}^2)$:

$$x_i^l \geq 0, \quad \text{for } i \in K_l ; \tag{3.1}$$

$$x_i^l \leq 1, \quad \text{for } i \in K_l . \tag{3.2}$$

We view the remaining inequalities that we present as lower and upper bounds on the product variable y . In the sequel, H is a subset of $K_l \times K_{\bar{l}}$, where l is either 1 or 2, and $\bar{l} = 3 - l$.

Consider the following *lower bound inequalities* for y :

$$y \geq \sum_{(i,j) \in H} a_i^1 a_j^2 (x_i^1 + x_j^2 - 1) . \tag{3.3}$$

PROPOSITION 3.1. *The inequalities (3.3) are valid for $P(\mathbf{a}^1, \mathbf{a}^2)$.*

Proof.

$$\begin{aligned} y &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^1 x_j^2 \\ &\geq \sum_{(i,j) \in H} a_i^1 a_j^2 x_i^1 x_j^2 \\ &\geq \sum_{(i,j) \in H} a_i^1 a_j^2 (x_i^1 + x_j^2 - 1) . \end{aligned}$$

□

We derive upper bounds on y by considering certain transformations ϕ^l , $l = 1, 2$, of the polytope $P(\mathbf{a}^1, \mathbf{a}^2)$, defined by the following formulae:

$$\begin{aligned}\tilde{x}_i^l &= 1 - x_i^l, \quad \text{for } i \in K_l; \\ \tilde{x}_j^{\bar{l}} &= x_j^{\bar{l}}, \quad \text{for } j \in K_{\bar{l}}; \\ \tilde{y} &= \sum_{i \in K_l} \sum_{j \in K_{\bar{l}}} a_i^l a_j^{\bar{l}} \tilde{x}_i^l \tilde{x}_j^{\bar{l}}.\end{aligned}$$

PROPOSITION 3.2. *The transformation ϕ^l , $l = 1, 2$, is an affine involution.*

Proof. Clearly $\phi^l \circ \phi^l$ is the identity transformation. To see that it is affine, we need just check that \tilde{y} is an affine function of the original variables.

$$\begin{aligned}\tilde{y} &= \sum_{i \in K_l} \sum_{j \in K_{\bar{l}}} a_i^l a_j^{\bar{l}} \tilde{x}_i^l \tilde{x}_j^{\bar{l}} \\ &= \sum_{i \in K_l} \sum_{j \in K_{\bar{l}}} a_i^l a_j^{\bar{l}} (1 - x_i^l) x_j^{\bar{l}} \\ &= \sum_{i \in K_l} \sum_{j \in K_{\bar{l}}} a_i^l a_j^{\bar{l}} x_j^{\bar{l}} - \sum_{i \in K_l} \sum_{j \in K_{\bar{l}}} a_i^l a_j^{\bar{l}} x_i^l x_j^{\bar{l}} \\ &= \sum_{i \in K_l} \sum_{j \in K_{\bar{l}}} a_i^l a_j^{\bar{l}} x_j^{\bar{l}} - y.\end{aligned}$$

□

Our *upper bound inequalities* take the form:

$$y \leq \sum_{i \in K_l} \sum_{j \in K_{\bar{l}}} a_i^l a_j^{\bar{l}} x_j^{\bar{l}} + \sum_{(i,j) \in H} a_i^l a_j^{\bar{l}} (x_i^l - x_j^{\bar{l}}). \quad (3.4)$$

PROPOSITION 3.3. *The inequalities (3.4) are valid for $P(\mathbf{a}^1, \mathbf{a}^2)$.*

Proof. We apply the lower bound inequalities (3.3) to the variables transformed by ϕ^l , to obtain

$$\sum_{i \in K_l} \sum_{j \in K_{\bar{l}}} a_i^l a_j^{\bar{l}} x_j^{\bar{l}} - y = \tilde{y} \geq \sum_{(i,j) \in H} a_i^l a_j^{\bar{l}} (\tilde{x}_i^l + \tilde{x}_j^{\bar{l}} - 1).$$

Solving for y , and rewriting this in terms of x^l and $x^{\bar{l}}$, we obtain the upper bound inequalities (3.4). □

Note that the sets of inequalities (3.4) with $l = 1$ and $l = 2$ are equivalent — this follows by checking that changing l is equivalent to complementing H .

The transformation ϕ^l corresponds to the “switching” operation used in the analysis of the cut polytope (see [DL97], for example). Specifically, (1.2) and (1.3) are switches of each other under ϕ^1 , and (1.2) and (1.4) are switches of each other under ϕ^2 .

PROPOSITION 3.4. *The points satisfying (1.6) and (3.3–3.4) for all cross products H are precisely the points satisfying (1.5–1.6).*

Proof. By Propositions 3.1 and 3.3, we need only show that every point satisfying (1.6) and (3.3–3.4) for all cross products H also satisfies (1.5). Let $(\mathbf{x}^1, \mathbf{x}^2, y)$ be a point satisfying (1.6) and (3.3–3.4). Letting

$$H = \{i \in K_1 : x_i^1 = 1\} \times \{j \in K_2 : x_j^2 = 1\},$$

we obtain a lower bound inequality (3.3) that is satisfied as an equation by the point $(\mathbf{x}^1, \mathbf{x}^2, y)$. Similarly, letting

$$H = \{i \in K_1 : x_i^1 = 0\} \times \{j \in K_2 : x_j^2 = 1\},$$

we obtain an upper bound inequality (3.4) that is satisfied as an equation by the point $(\mathbf{x}^1, \mathbf{x}^2, y)$. So \mathbf{x}^1 and \mathbf{x}^2 together with (3.3–3.4) for cross products H determine y . But by the definition of $P(\mathbf{a}^1, \mathbf{a}^2)$, the point $(\mathbf{x}^1, \mathbf{x}^2, \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^1 x_j^2)$ is in $P(\mathbf{a}^1, \mathbf{a}^2)$, so (1.5) must be satisfied. \square

4. Facets. For $i \in K_l$, $l = 1, 2$, we let $\mathbf{e}^i \in \mathbb{R}^{k_l}$ denote the i -th standard unit vector. For simplicity, we say that a point is *tight* for an inequality if it satisfies the inequality as an equation.

PROPOSITION 4.1. *For $k_1, k_2 > 0$, the polytope $P(\mathbf{a}^1, \mathbf{a}^2)$ is full dimensional.*

Proof. We prove this directly. The following $k_1 + k_2 + 2$ points in $P(\mathbf{a}^1, \mathbf{a}^2)$ are easily seen to be affinely independent:

- $(\mathbf{x}^1, \mathbf{x}^2, y) = (\mathbf{0}, \mathbf{0}, 0)$;
- $(\mathbf{x}^1, \mathbf{x}^2, y) = (\mathbf{e}^j, \mathbf{0}, 0)$, for $j \in K_1$;
- $(\mathbf{x}^1, \mathbf{x}^2, y) = (\mathbf{0}, \mathbf{e}^j, 0)$, for $j \in K_2$;
- $(\mathbf{x}^1, \mathbf{x}^2, y) = (\mathbf{e}^1, \mathbf{e}^1, a_1^1 a_1^2)$.

\square

PROPOSITION 4.2. *For $l = 1, 2$, the inequalities (3.1) describe facets of $P(\mathbf{a}^1, \mathbf{a}^2)$ when $k_l > 1$.*

Proof. Again, we proceed directly. For both $l = 1$ and $l = 2$, the following $k_1 + k_2 + 1$ points in $P(\mathbf{a}^1, \mathbf{a}^2)$ are tight for (3.1) and are easily seen to be affinely independent:

- $(\mathbf{x}^1, \mathbf{x}^2, y) = (\mathbf{0}, \mathbf{0}, 0)$;
- $(\mathbf{x}^1, \mathbf{x}^2, y)$ defined by $\mathbf{x}^l = \mathbf{e}^j$, $\mathbf{x}^{\bar{l}} = \mathbf{0}$, $y = 0$, for $j \in K_l \setminus \{i\}$;
- $(\mathbf{x}^1, \mathbf{x}^2, y)$ defined by $\mathbf{x}^l = \mathbf{0}$, $\mathbf{x}^{\bar{l}} = \mathbf{e}^j$, $y = 0$, for $j \in K_{\bar{l}}$;
- $(\mathbf{x}^1, \mathbf{x}^2, y)$ defined by $\mathbf{x}^l = \mathbf{e}^m$, for some $m \in K_l \setminus \{i\}$, $\mathbf{x}^{\bar{l}} = \mathbf{e}^1$, $y = a_m^l a_1^{\bar{l}}$.

\square

PROPOSITION 4.3. *For $l = 1, 2$, the inequalities (3.2) describe facets of $P(\mathbf{a}^1, \mathbf{a}^2)$ when $k_l > 1$.*

Proof. The affine transformation ϕ^l is an involution from the face of $P(\mathbf{a}^1, \mathbf{a}^2)$ described by the simple lower bound inequality $x_i^l \geq 0$ to the face of $P(\mathbf{a}^1, \mathbf{a}^2)$ described by the simple upper bound inequality $x_i^l \leq 1$.

Since the affine transformation is invertible, it is dimension preserving. Therefore, by Proposition 4.2, the result follows. \square

One might suspect from Proposition 3.4 that the only inequalities of the form (3.3–3.4) that yield facets have H being a cross product, but this is not the case. We now define a key subset $\mathcal{H}(k_1, k_2)$ of the set of subsets of $K_1 \times K_2$. Each H in $\mathcal{H}(k_1, k_2)$ arises by choosing a permutation of the variables

$$\{x_i^1 : i \in K_1\} \cup \{x_j^2 : j \in K_2\}.$$

If x_i^1 precedes x_j^2 in the permutation, which we denote by $x_i^1 \prec x_j^2$, then (i, j) is in H .

For example, if $S_1 \subset K_1$ and $S_2 \subset K_2$, then we get the cross product $H = S_1 \times S_2 \in \mathcal{H}(k_1, k_2)$ by choosing any permutation of the variables having $\{x_j^2 : j \in K_2 \setminus S_2\}$ first, followed by $\{x_i^1 : i \in S_1\}$, followed by $\{x_j^2 : j \in S_2\}$, followed by $\{x_i^1 : i \in K_1 \setminus S_1\}$.

As another example, let $k_1 = 3$ and $k_2 = 2$, and consider the permutation: $x_2^1, x_2^2, x_1^1, x_3^1, x_1^2$, which yields

$$H = \{(2, 2), (2, 1), (1, 1), (3, 1)\}.$$

This choice of H is not a cross product. However, this H yields the lower bound inequality:

$$\begin{aligned} y \geq & a_2^1 a_2^2 (x_2^1 + x_2^2 - 1) \\ & + a_2^1 a_1^2 (x_2^1 + x_1^2 - 1) \\ & + a_1^1 a_1^2 (x_1^1 + x_1^2 - 1) \\ & + a_3^1 a_1^2 (x_3^1 + x_1^2 - 1). \end{aligned}$$

We demonstrate that this inequality describes a facet by displaying $k_1 + k_2 + 1 = 6$ affinely independent points in $P(\mathbf{a}^1, \mathbf{a}^2)$ that are tight for the inequality: We display the points as rows of the matrix below, where we have permuted the columns, in a certain manner, according to the permutation of the variables that led to H , and we have inserted a column of 1's. It suffices to check that the square “caterpillar matrix” obtained by deleting the y column is nonsingular.

x_2^2	x_1^2	1	x_2^1	x_1^1	x_3^1	y
0	0	1	1	1	1	$0 = (a_3^1 + a_1^1 + a_2^1)(0)$
0	1	1	1	1	1	$(a_3^1 + a_1^1 + a_2^1)(a_1^2)$
0	1	1	1	1	0	$(a_1^1 + a_2^1)(a_1^2)$
0	1	1	1	0	0	$(a_2^1)(a_1^2)$
1	1	1	1	0	0	$(a_2^1)(a_1^2 + a_2^2)$
1	1	1	0	0	0	$0 = (0)(a_1^2 + a_2^2)$

For $m \geq 3$, an order- m *caterpillar matrix* is obtained by choosing $k_1, k_2 \geq 1$ with $k_1 + k_2 + 1 = m$. The first row of such a matrix is $(\mathbf{0} \in \mathbb{R}^{k_2}, 1, \mathbf{1} \in \mathbb{R}^{k_1})$, and the last row is $(\mathbf{1} \in \mathbb{R}^{k_2}, 1, \mathbf{0} \in \mathbb{R}^{k_1})$. Each row is obtained from the one above it by either flipping the right-most 1 to 0, or flipping the 0 immediately to the left of the left-most 1 to 1. So the rows depict snapshots of a “caterpillar” of 1’s that moves from right to left by either pushing its head forward a bit or pulling its tail forward a bit. We note that besides translating affine to linear dependence, the column of 1’s precludes the unsettling possibility of a 1-bit caterpillar disappearing by pulling its tail forward a bit, and then reappearing by pushing its head forward a bit. Since caterpillar matrices have their 1’s in each row consecutively appearing, they are totally unimodular (see [HK56]). That is, the determinant of a caterpillar matrix is in $\{0, \pm 1\}$. In fact, we have the following result.

PROPOSITION 4.4. *The determinant of a caterpillar matrix is in $\{\pm 1\}$.*

Proof. The proof is by induction on m . The only order-3 caterpillar matrices are

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

It is easy to check that these have determinant in $\{\pm 1\}$.

Now, suppose that we have a caterpillar matrix of order $m \geq 4$. Depending on the bit flip that produces the last row from the one above it, the matrix has the form

$$\left(\begin{array}{cccc|ccccc} 0 & 0 & \cdots & 0 & 1 & 1 & 1 & \cdots & 1 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 & \cdots & 1 & 0 \\ & & & & \vdots & & & & \vdots & \\ & & & & 1 & & & & 0 & \\ & & & & 1 & & & & 0 & \end{array} \right)$$

or

$$\left(\begin{array}{cccc|ccccc} 0 & 0 & \cdots & 0 & 0 & 1 & 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 & 1 & \cdots & 1 \\ & & & & \vdots & \vdots & & & & \\ & & & & 1 & 1 & & & & \\ & & & & 1 & 1 & & & & \end{array} \right).$$

In the first case, we can expand the determinant along the last column and we obtain a caterpillar matrix of order $m - 1$ with the same determinant as the original matrix, up to the sign. In the second case, we subtract the first row from the second row (which does not affect the determinant), and

then we expand along the second row of the resulting matrix. Again, we obtain a caterpillar matrix of order $m - 1$ with the same determinant as the original matrix, up to the sign. In either case, the result follows by induction. \square

PROPOSITION 4.5. *An inequality of the form (3.3) describes a facet of $P(\mathbf{a}^1, \mathbf{a}^2)$ when $H \in \mathcal{H}(k_1, k_2)$.*

Proof. By Proposition 3.1, these inequalities are valid for $P(\mathbf{a}^1, \mathbf{a}^2)$. It suffices to exhibit $k_1 + k_2 + 1$ affinely independent points of $P(\mathbf{a}^1, \mathbf{a}^2)$ that are tight for (3.3). Let Φ be the permutation that gives rise to H . It is easy to check that $(\mathbf{x}^1, \mathbf{x}^2, y) = (\mathbf{0}, \mathbf{1}, 0)$ is a point of $P(\mathbf{a}^1, \mathbf{a}^2)$ that is tight for (3.3). We generate the remaining $k_1 + k_2$ points by successively flipping bits in the order of the permutation Φ . We simply need to check that each bit flip preserves equality in (3.3). If a variable x_i^1 is flipped from 0 to 1, the increase in y (i.e., the left-hand side of (3.3)) and in $\sum_{(i,j) \in H} a_i^1 a_j^2 (x_i^1 + x_j^2 - 1)$ (i.e., the right-hand side of (3.3)) is precisely $\sum_{j: x_i^1 \prec x_j^2} a_i^1 a_j^2$. Similarly, if a variable x_j^2 is flipped from 1 to 0, the decrease in both of these quantities is precisely $\sum_{i: x_i^1 \prec x_j^2} a_i^1 a_j^2$.

Next, we arrange these $k_1 + k_2 + 1$ points, in the order generated, as the rows of a caterpillar matrix of order $k_1 + k_2 + 1$. A point $(\mathbf{x}^1, \mathbf{x}^2, y)$ yields the row $(\mathbf{x}_\Phi^2, 1, \mathbf{x}_\Phi^1)$, where \mathbf{x}_Φ^l is just \mathbf{x}^l permuted according to the order of the x_i^l in Φ . Clearly this defines a caterpillar matrix, which is nonsingular by Proposition 4.4. Hence, the generated points are affinely independent, so (3.3) describes a facet when $H \in \mathcal{H}(k_1, k_2)$. \square

COROLLARY 4.1. *Each inequality (3.3) with $H \in \mathcal{H}(k_1, k_2)$ admits a set of tight points in $P(\mathbf{a}^1, \mathbf{a}^2)$ that correspond to the rows of a caterpillar matrix.*

PROPOSITION 4.6. *An inequality of the form (3.4) describes a facet of $P(\mathbf{a}^1, \mathbf{a}^2)$ when $H \in \mathcal{H}(k_1, k_2)$.*

Proof. Using the transformation ϕ^l , this follows from Proposition 4.5. \square

Conversely, every caterpillar matrix of order $k_1 + k_2 + 1$ corresponds to a facet of the form (3.3). More precisely, we have the following result.

PROPOSITION 4.7. *Let C be a caterpillar matrix of order $k_1 + k_2 + 1$ such that its first k_2 columns correspond to a specific permutation of $\{x_j^2 : j \in K_2\}$ and its last k_1 columns correspond to a specific permutation of $\{x_i^1 : i \in K_1\}$. Then there exists a facet of $P(a_1, a_2)$ of the form (3.3) such that the points corresponding to the rows of C are tight for it.*

Proof. It is easy to determine the permutation Ψ that corresponds to C , by interleaving the given permutations of $\{x_j^2 : j \in K_2\}$ and of $\{x_i^1 : i \in K_1\}$, according to the head and tail moves of the caterpillar. Then, as before, we form the H of (3.3) by putting (i, j) in H if $x_i^1 \prec x_j^2$ in the final permutation.

It is easy to see that each row of C corresponds to a point of $P(\mathbf{a}^1, \mathbf{a}^2)$ that is tight for the resulting inequality (3.3). \square

5. Inequality characterization. In this section, we demonstrate how every facet of $P(\mathbf{a}^1, \mathbf{a}^2)$ is one of the ones described in Section 4. We do this by projecting from an extended formulation in a higher-dimensional space. Of course this is a well-known technique (see e.g., [BP83, BP89, Bal01]).

Consider the system

$$y = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \delta_{ij} ; \quad (5.1)$$

$$\delta_{ij} \leq x_i^1 , \text{ for all } i \in K_1, j \in K_2 ; \quad (5.2)$$

$$\delta_{ij} \leq x_j^2 , \text{ for all } i \in K_1, j \in K_2 ; \quad (5.3)$$

$$\delta_{ij} \geq x_i^1 + x_j^2 - 1 , \text{ for all } i \in K_1, j \in K_2 ; \quad (5.4)$$

$$\delta_{ij} \geq 0 , \text{ for all } i \in K_1, j \in K_2 , \quad (5.5)$$

and let

$$Q^\delta(\mathbf{a}^1, \mathbf{a}^2) = \text{conv} \left\{ \mathbf{x}^1 \in \mathbb{R}^{k_1}, \mathbf{x}^2 \in \mathbb{R}^{k_2}, y \in \mathbb{R}, \delta \in \mathbb{R}^{k_1 \times k_2} : (3.1-3.2, 5.1-5.5) \right\},$$

where we use $\text{conv}(X)$ to denote the convex hull of X . Let $Q(\mathbf{a}^1, \mathbf{a}^2)$ be the projection of $Q^\delta(\mathbf{a}^1, \mathbf{a}^2)$ in the space of $\mathbf{x}^1, \mathbf{x}^2$ and y variables. We next show that $Q(\mathbf{a}^1, \mathbf{a}^2)$ is integral.

PROPOSITION 5.1. $Q(\mathbf{a}^1, \mathbf{a}^2)$ is integral on \mathbf{x}^1 and \mathbf{x}^2 .

Proof. We will show that if \mathbf{p} is fractional (on \mathbf{x}^1 and \mathbf{x}^2), then it is not an extreme point of $Q(\mathbf{a}^1, \mathbf{a}^2)$.

Assume that $\mathbf{p} = (\mathbf{x}^1, \mathbf{x}^2, y)$ is a fractional extreme point of $Q(\mathbf{a}^1, \mathbf{a}^2)$. For $v \in \mathbb{R}$, let $(v)^+ = \max\{0, v\}$. For fixed \mathbf{x}^1 and \mathbf{x}^2 , notice that $\delta \in \mathbb{R}^{k_1 \times k_2}$ is feasible to (5.2-5.5) if and only if it satisfies $\min\{x_i^1, x_j^2\} \geq \delta_{ij} \geq (x_i^1 + x_j^2 - 1)^+$ for all $i \in K_1, j \in K_2$. Therefore, if we define

$$y_{up} = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \min\{x_i^1, x_j^2\},$$

$$y_{down} = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 (x_i^1 + x_j^2 - 1)^+$$

then the points $\mathbf{p}_{up} = (\mathbf{x}^1, \mathbf{x}^2, y_{up})$, and $\mathbf{p}_{down} = (\mathbf{x}^1, \mathbf{x}^2, y_{down})$ are in $Q(\mathbf{a}^1, \mathbf{a}^2)$, and $\mathbf{p}_{up} \geq \mathbf{p} \geq \mathbf{p}_{down}$. Furthermore, if \mathbf{p} is an extreme point, it has to be one of \mathbf{p}_{up} and \mathbf{p}_{down} .

Let $\bar{K}_1 \subseteq K_1$ and $\bar{K}_2 \subseteq K_2$ be the set of indices corresponding to fractional components of \mathbf{x}^1 and \mathbf{x}^2 respectively. Clearly, $\bar{K}_1 \cup \bar{K}_2 \neq \emptyset$. Let

$\epsilon > 0$ be a small number so that $1 > x_i^l + \epsilon > x_i^l - \epsilon > 0$ for all $i \in \bar{K}_l$, $l = 1, 2$. Define \mathbf{x}^{l+} where $x_i^{l+} := x_i^l + \epsilon$ if $i \in \bar{K}_l$ and $x_i^{l+} := x_i^l$, otherwise. Define \mathbf{x}^{l-} similarly. We consider the following two cases and show that if \mathbf{p} is fractional, then it can be represented as a convex combination of two distinct points in $Q(\mathbf{a}^1, \mathbf{a}^2)$.

Case 1. Assume $\mathbf{p} = \mathbf{p}_{up}$. Let $\mathbf{p}^a = (\mathbf{x}^{1+}, \mathbf{x}^{2+}, y^a)$ and $\mathbf{p}^b = (\mathbf{x}^{1-}, \mathbf{x}^{2-}, y^b)$ where

$$\begin{aligned} y^a &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \min\{x_i^{1+}, x_j^{2+}\}, \\ y^b &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \min\{x_i^{1-}, x_j^{2-}\}, \end{aligned}$$

and note that $\mathbf{p}^a, \mathbf{p}^b \in Q(\mathbf{a}^1, \mathbf{a}^2)$ and $\mathbf{p}^a \neq \mathbf{p}^b$.

For $i \in K_1$ and $j \in K_2$, let $\delta_{ij} = \min\{x_i^1, x_j^2\}$, and define δ_{ij}^a and δ_{ij}^b similarly. Note that $y = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \delta_{ij}$. Due to the construction, if $\min\{x_i^1, x_j^2\} = 0$, then we have $\min\{x_i^{1+}, x_j^{2+}\} = \min\{x_i^{1-}, x_j^{2-}\} = 0$, and therefore $\delta_{ij} = \delta_{ij}^a = \delta_{ij}^b = 0$. Similarly, if $\delta_{ij} = 1$, then we have $\delta_{ij}^a = \delta_{ij}^b = 1$ as well. On the other hand, if $\delta_{ij} \notin \{0, 1\}$, then $\delta_{ij}^a = \delta_{ij} + \epsilon$ and $\delta_{ij}^b = \delta_{ij} - \epsilon$. Therefore, $\delta^a + \delta^b = 2\delta$ and $\mathbf{p} = (1/2)\mathbf{p}^a + (1/2)\mathbf{p}^b$.

Case 2. Assume $\mathbf{p} = \mathbf{p}_{down}$. Let $\mathbf{p}^c = (\mathbf{x}^{1+}, \mathbf{x}^{2-}, y^c)$ and $\mathbf{p}^d = (\mathbf{x}^{1-}, \mathbf{x}^{2+}, y^d)$ where

$$\begin{aligned} y^c &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 (x_i^{1+} + x_j^{2-} - 1)^+, \\ y^d &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 (x_i^{1-} + x_j^{2+} - 1)^+, \end{aligned}$$

and note that $\mathbf{p}^c, \mathbf{p}^d \in Q(\mathbf{a}^1, \mathbf{a}^2)$ and $\mathbf{p}^c \neq \mathbf{p}^d$.

Let $\delta_{ij} = (x_i^1 + x_j^2 - 1)^+$ for $i \in K_1$ and $j \in K_2$, and define δ_{ij}^c and δ_{ij}^d similarly. Note that $y = \sum_{i \in K_1} \sum_{j \in K_2} \delta_{ij}$.

If $\min\{x_i^1, x_j^2\} = 0$, then $\min\{x_i^{1+}, x_j^{2-}\} = \min\{x_i^{1-}, x_j^{2+}\} = 0$ and $\delta_{ij} = \delta_{ij}^c = \delta_{ij}^d = 0$. If $x_i^1 = 1$, then $\delta_{ij} = x_j^2$, implying $\delta_{ij}^c = x_j^{2-}$ and $\delta_{ij}^d = x_j^{2+}$. Similarly, if $x_i^2 = 1$, then $\delta_{ij} = x_i^1$, implying $\delta_{ij}^c = x_i^{1+}$ and $\delta_{ij}^d = x_i^{1-}$. Finally, if $1 > x_i^1, x_j^2 > 0$, then $\delta_{ij} = \delta_{ij}^c = \delta_{ij}^d$. Therefore, $\delta^c + \delta^d = 2\delta$ and $\mathbf{p} = (1/2)\mathbf{p}^c + (1/2)\mathbf{p}^d$. \square

Now, let $R(\mathbf{a}^1, \mathbf{a}^2)$ be the real solution set of (3.1–3.4), and note that $P(\mathbf{a}^1, \mathbf{a}^2) \subseteq R(\mathbf{a}^1, \mathbf{a}^2)$. To prove that $P(\mathbf{a}^1, \mathbf{a}^2) = R(\mathbf{a}^1, \mathbf{a}^2)$, we will first argue that $Q(\mathbf{a}^1, \mathbf{a}^2) \subseteq P(\mathbf{a}^1, \mathbf{a}^2)$, and then we will show that $R(\mathbf{a}^1, \mathbf{a}^2) \subseteq Q(\mathbf{a}^1, \mathbf{a}^2)$.

PROPOSITION 5.2. $Q(\mathbf{a}^1, \mathbf{a}^2) \subseteq P(\mathbf{a}^1, \mathbf{a}^2)$.

Proof. As $P(\mathbf{a}^1, \mathbf{a}^2)$ is a bounded convex set, it is sufficient to show that all of the extreme points of $Q(\mathbf{a}^1, \mathbf{a}^2)$ are contained in $P(\mathbf{a}^1, \mathbf{a}^2)$. Using

Proposition 5.1 and its proof, we know that if $\mathbf{p} = (\mathbf{x}^1, \mathbf{x}^2, y)$ is an extreme point of $Q(\mathbf{a}^1, \mathbf{a}^2)$, then \mathbf{x}^1 and \mathbf{x}^2 are integral and

$$\sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \min\{x_i^1, x_j^2\} \geq y \geq \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 (x_i^1 + x_j^2 - 1)^+.$$

Notice that for any $u, v \in \{0, 1\}$, $\min\{u, v\} = (u+v-1)^+ = uv$. Therefore, $y = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^1 x_j^2$, and $\mathbf{p} \in P(\mathbf{a}^1, \mathbf{a}^2)$. \square

PROPOSITION 5.3. $R(\mathbf{a}^1, \mathbf{a}^2) \subseteq Q(\mathbf{a}^1, \mathbf{a}^2)$.

Proof. Assume not, and let $\mathbf{p} = (\mathbf{x}^1, \mathbf{x}^2, y) \in R(\mathbf{a}^1, \mathbf{a}^2) \setminus Q(\mathbf{a}^1, \mathbf{a}^2)$. As in the proof of Proposition 5.1, let $\mathbf{p}_{up} = (\mathbf{x}^1, \mathbf{x}^2, y_{up})$ and $\mathbf{p}_{down} = (\mathbf{x}^1, \mathbf{x}^2, y_{down})$, where $y_{up} = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \min\{x_i^1, x_j^2\}$ and $y_{down} = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 (x_i^1 + x_j^2 - 1)^+$. Note that, $\mathbf{p}_{up}, \mathbf{p}_{down} \in Q(\mathbf{a}^1, \mathbf{a}^2)$. We next show that $y_{up} \geq y \geq y_{down}$, and therefore $\mathbf{p} \in \text{conv}\{\mathbf{p}_{up}, \mathbf{p}_{down}\} \subseteq Q(\mathbf{a}^1, \mathbf{a}^2)$.

Let $H_1 = \{(i, j) \in K_1 \times K_2 : x_i^1 > x_j^2\}$ and $H_2 = \{(i, j) \in K_1 \times K_2 : x_j^2 + x_i^1 > 1\}$. Applying (3.4) with $H = H_1$ gives

$$\begin{aligned} y &\leq \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^1 + \sum_{(i, j) \in H_1} a_i^1 a_j^2 (x_j^2 - x_i^1) \\ &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^1 + \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \min\{0, x_j^2 - x_i^1\} \\ &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 (x_i^1 + \min\{0, x_j^2 - x_i^1\}) \\ &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \min\{x_i^1, x_j^2\} = y_{up}. \end{aligned}$$

Applying (3.3) with $H = H_2$ gives

$$\begin{aligned} y &\geq \sum_{(i, j) \in H_2} a_i^1 a_j^2 (x_i^1 + x_j^2 - 1) \\ &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \max\{0, x_i^1 + x_j^2 - 1\} = y_{down}. \end{aligned}$$

\square

As a consequence, we have our main theorem:

THEOREM 5.4. $P(\mathbf{a}^1, \mathbf{a}^2) = R(\mathbf{a}^1, \mathbf{a}^2) = Q(\mathbf{a}^1, \mathbf{a}^2)$.

Although our main goal was to establish the inequality description (3.1-3.4) of $P(\mathbf{a}^1, \mathbf{a}^2)$, we have established that from a mathematical point of view, the extended formulation (5.1-5.5) has the same power as a description. Which formulation will be preferable in an application will likely depend on implementation details.

6. Separation. We can efficiently include all facet describing inequalities of $P(\mathbf{a}^1, \mathbf{a}^2)$ implicitly in a linear programming formulation, provided that we can separate on them in polynomial time (see [GLS84, GLS81, GLS93]). That is, provided we have a polynomial-time algorithm that determines whether a given point is in $P(\mathbf{a}^1, \mathbf{a}^2)$ and provides a violated facet describing inequality if the point is not in $P(\mathbf{a}^1, \mathbf{a}^2)$.

Separation for the simple lower and upper bound inequalities (3.1–3.2) is easily handled by enumeration. For a point $(\mathbf{x}^1, \mathbf{x}^2, y)$ satisfying (3.1–3.2), separation for the lower and upper bound inequalities (3.3–3.4) is also rather simple. For the lower bound inequalities (3.3), we simply let

$$H_0 = \{(i, j) \in K_1 \times K_2 : x_i^1 + x_j^2 > 1\},$$

and then we just check whether $(\mathbf{x}^1, \mathbf{x}^2, y)$ violates the lower bound inequality (3.3) for the choice of $H = H_0$. Similarly, for the upper bound inequalities (3.4), we let

$$H_0 = \{(i, j) \in K_l \times K_{\bar{l}} : x_i^l - x_j^{\bar{l}} < 0\},$$

and then we just check whether $(\mathbf{x}^1, \mathbf{x}^2, y)$ violates the upper bound inequality (3.4) for the choice of $H = H_0$. Note that for any $H \subseteq K_1 \times K_2$,

$$\sum_{(i,j) \in H} a_i^1 a_j^2 (x_i^1 + x_j^2 - 1) \leq \sum_{(i,j) \in H_0} a_i^1 a_j^2 (x_i^1 + x_j^2 - 1).$$

Therefore, $(\mathbf{x}^1, \mathbf{x}^2, y)$ satisfies the lower bounds (3.3) for all sets $H \subseteq K_1 \times K_2$ if and only if it satisfies (3.3) for $H = H_0$.

Using Propositions 4.5 and 4.6, we can see how this separation method yields *facet describing* violated inequalities. We develop a permutation of the variables

$$\{x_i^1 : i \in K_1\} \cup \{x_j^2 : j \in K_2\},$$

according to their values. Let $\delta_1 > \delta_2 > \dots > \delta_p$ denote the distinct values in $\{x_i^1 : i \in K_1\}$. For convenience, let $\delta_0 = 1$ and $\delta_{p+1} = 0$. We define the partition via $2p + 1$ blocks, some of which may be empty. For, $t = 1, 2, \dots, p$, block $2t$ consists of

$$\{x_i^1 : i \in K_1, x_i^1 = \delta_t\}.$$

For, $t = 1, 2, \dots, p$, block $2t + 1$ consists of

$$\{x_j^2 : j \in K_2, 1 - \delta_t < x_j^2 \leq 1 - \delta_{t+1}\},$$

and block 1 consists of

$$\{x_j^2 : j \in K_2, 1 - \delta_0 = 0 \leq x_j^2 \leq 1 - \delta_{t+1}\}.$$

This permutation of the variables determines a subset H of $K_1 \times K_2$ as described in Section 4. This choice of H yields a facet-describing lower-bound inequality (3.3).

Similarly, for the upper bound inequalities (3.4), we let $\delta_1 < \delta_2 < \dots < \delta_p$ denote the distinct values of the $\{x_i^l : i \in K_l\}$. As before, let $\delta_0 = 0$ and $\delta_{p+1} = 1$, and we define a partition via $2p + 1$ blocks, some of which may be empty. For, $t = 1, 2, \dots, p$, block $2t$ consists of

$$\{x_i^l : i \in K_l, x_i^l = \delta_t\}.$$

For, $t = 0, 1, 2, \dots, p$, block $2t + 1$ consists of

$$\{x_j^{\bar{l}} : j \in K_{\bar{l}}, \delta_t < x_j^{\bar{l}} \leq \delta_{t+1}\}.$$

This permutation of the variables determines a subset H of $K_1 \times K_2$ as described in Section 4. This choice of H yields a facet describing upper bound inequality (3.4).

Our separation algorithm can easily be implemented with running time $\mathcal{O}(k_1 k_2)$.

7. Ideal points. For many combinatorial polytopes, it is natural to investigate adjacency of extreme points via edges (i.e., 1-dimensional faces). One motivation is that this notion of adjacency may prove useful in some local-search heuristics. In this section, we investigate a different notion of adjacency for extreme points — one that seems more natural for $P(\mathbf{a}^1, \mathbf{a}^2)$ as it directly captures the product structure.

The point $(\mathbf{x}^1, \mathbf{x}^2, y) \in P(\mathbf{a}^1, \mathbf{a}^2)$ is *ideal* if it satisfies (1.5). Clearly, the extreme points of $P(\mathbf{a}^1, \mathbf{a}^2)$ are ideal. Also, $P(\mathbf{a}^1, \mathbf{a}^2)$ contains points that are not ideal. For example,

$$(x_1^1, x_1^2, y) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}) = \frac{1}{2}(0, 0, 0) + \frac{1}{2}(1, 1, 1)$$

is in $P((1), (1))$, but it is not ideal because

$$\sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^1 x_j^2 = 1 \cdot 1 \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \neq \frac{1}{2}.$$

PROPOSITION 7.1. *Every pair of distinct extreme points of $P(\mathbf{a}^1, \mathbf{a}^2)$ is connected by a curve of ideal points of $P(\mathbf{a}^1, \mathbf{a}^2)$. Moreover, such a curve can be taken to be either a line segment or two line segments joined at another extreme point of $P(\mathbf{a}^1, \mathbf{a}^2)$.*

Proof. Let $(\mathbf{x}^{11}, \mathbf{x}^{12}, y^{11})$ and $(\mathbf{x}^{21}, \mathbf{x}^{22}, y^{22})$ be a pair of distinct extreme points of $P(\mathbf{a}^1, \mathbf{a}^2)$. If $\mathbf{x}^{11} = \mathbf{x}^{21}$ then we consider the curve obtained by letting

$$(\mathbf{z}^1, \mathbf{z}^2, y) = \lambda(\mathbf{x}^{11}, \mathbf{x}^{12}, y^{11}) + (1 - \lambda)(\mathbf{x}^{21}, \mathbf{x}^{22}, y^{22}),$$

as λ ranges between 0 and 1. For $\lambda = 1$ we have $(\mathbf{z}^1, \mathbf{z}^2, y) = (\mathbf{x}^{11}, \mathbf{x}^{12}, y^{11})$, and for $\lambda = 0$ we have $(\mathbf{z}^1, \mathbf{z}^2, y) = (\mathbf{x}^{21}, \mathbf{x}^{22}, y^{22})$. Clearly, the curve is a line segment entirely contained in the convex polytope $P(\mathbf{a}^1, \mathbf{a}^2)$, because we have defined each point on the curve as a convex combination of the pair of extreme points of P . So it remains to demonstrate that each point on the curve is ideal:

$$\begin{aligned} & \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 z_i^1 z_j^2 \\ &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 (\lambda x_i^{11} + (1 - \lambda)x_i^{21}) \cdot (\lambda x_j^{12} + (1 - \lambda)x_j^{22}) \\ &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^{11} (\lambda x_j^{12} + (1 - \lambda)x_j^{22}) \\ &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 \lambda x_i^{11} x_j^{12} + (1 - \lambda)x_i^{21} x_j^{22} \\ &= \lambda y^{11} + (1 - \lambda)y^{22} \\ &= y . \end{aligned}$$

Therefore the points on the curve are ideal.

Similarly, if $\mathbf{x}^{12} = \mathbf{x}^{22}$, we use the same line segment above to connect the points.

Suppose now that $\mathbf{x}^{11} \neq \mathbf{x}^{21}$ and $\mathbf{x}^{12} \neq \mathbf{x}^{22}$. We define a third point (x^{11}, x^{22}, y^{12}) , where

$$y^{12} = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^{11} x_j^{22} .$$

Then we connect this third point, in the manner above, to each of the points of the original pair. \square

The curve of ideal points given to us by Proposition 7.1 is entirely contained in a 2-dimensional polytope, but it is not smooth in general. By allowing the curve to be contained in a 3-dimensional polytope, we can construct a smooth curve of ideal points connecting each pair of extreme points of $P(\mathbf{a}^1, \mathbf{a}^2)$.

PROPOSITION 7.2. *Every pair of extreme points of $P(\mathbf{a}^1, \mathbf{a}^2)$ is connected by a smooth curve of ideal points of $P(\mathbf{a}^1, \mathbf{a}^2)$.*

Proof. Let $(\mathbf{x}^{11}, \mathbf{x}^{12}, y^{11})$ and $(\mathbf{x}^{21}, \mathbf{x}^{22}, y^{22})$ be a pair of distinct extreme points of $P(\mathbf{a}^1, \mathbf{a}^2)$. Our goal is to connect these points with a smooth curve of ideal points of $P(\mathbf{a}^1, \mathbf{a}^2)$. Toward this end, we consider two other points of $P(\mathbf{a}^1, \mathbf{a}^2)$, $(\mathbf{x}^{11}, \mathbf{x}^{22}, y^{12})$ and $(\mathbf{x}^{21}, \mathbf{x}^{12}, y^{21})$, which we obtain from the original pair by letting

$$y^{12} = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^{11} x_j^{22}$$

and

$$y^{21} = \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^{21} x_j^{12} .$$

Now, we consider the curve obtained by letting

$$\begin{aligned} (\mathbf{z}^1, \mathbf{z}^2, y) &= \lambda^2(\mathbf{x}^{11}, \mathbf{x}^{12}, y^{11}) + (1 - \lambda)^2(\mathbf{x}^{21}, \mathbf{x}^{22}, y^{22}) \\ &\quad + \lambda(1 - \lambda)(\mathbf{x}^{11}, \mathbf{x}^{22}, y^{12}) + \lambda(1 - \lambda)(\mathbf{x}^{21}, \mathbf{x}^{12}, y^{21}) , \end{aligned}$$

as λ ranges between 0 and 1. For $\lambda = 1$ we have $(\mathbf{z}^1, \mathbf{z}^2, y) = (\mathbf{x}^{11}, \mathbf{x}^{12}, y^{11})$, and for $\lambda = 0$ we have $(\mathbf{z}^1, \mathbf{z}^2, y) = (\mathbf{x}^{21}, \mathbf{x}^{22}, y^{22})$. Clearly, the curve is entirely contained in the convex polytope $P(\mathbf{a}^1, \mathbf{a}^2)$, because we have defined each point on the curve as a convex combination of extreme points of P . So it remains to demonstrate that each point on the curve is ideal:

$$\begin{aligned} &\sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 z_i^1 z_j^2 \\ &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 (\lambda^2 x_i^{11} + (1 - \lambda)^2 x_i^{21} + \lambda(1 - \lambda)x_i^{11} + \lambda(1 - \lambda)x_i^{21}) \\ &\quad \cdot (\lambda^2 x_j^{12} + (1 - \lambda)^2 x_j^{22} + \lambda(1 - \lambda)x_j^{22} + \lambda(1 - \lambda)x_j^{12}) \\ &= \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 (\lambda x_i^{11} + (1 - \lambda)x_i^{21}) \cdot (\lambda x_j^{12} + (1 - \lambda)x_j^{22}) \\ &= \lambda^2 \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^{11} x_j^{12} + (1 - \lambda)^2 \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^{21} x_j^{22} \\ &\quad + \lambda(1 - \lambda) \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^{11} x_j^{22} + \lambda(1 - \lambda) \sum_{i \in K_1} \sum_{j \in K_2} a_i^1 a_j^2 x_i^{21} x_j^{12} \\ &= \lambda^2 y^{11} + (1 - \lambda)^2 y^{22} + \lambda(1 - \lambda)y^{12} + \lambda(1 - \lambda)y^{21} \\ &= y . \end{aligned}$$

Therefore the points on the curve are ideal. \square

8. A generalization. Although we do not have an application for this, our results generalize. Let \mathbf{A} be a $k_1 \times k_2$ matrix with positive components, and let $P(\mathbf{A})$ be the convex hull of solutions of

$$\begin{aligned} y &= \sum_{i \in K_1} \sum_{j \in K_2} a_{ij} x_i^1 x_j^2 ; \\ x_i^l &\in \{0, 1\}, \text{ for } i \in K_l, l = 1, 2 . \end{aligned}$$

The reader can easily check that everything that we have done applies to $P(\mathbf{A})$ by making the substitution of $a_i^1 a_j^2$ by a_{ij} throughout.

Acknowledgments. The authors are extremely grateful to Don Coppersmith who was a key participant in earlier versions of this work (see [CGLL03, CLL99]). The authors are grateful to Komei Fukuda for making his program *cdd* available. Evidence collected with the use of *cdd* led us to conjecture some of our results. The research of Jon Lee was supported in part by the Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong. The work of Janny Leung was partially supported by the Hong Kong Research Grants Council.

REFERENCES

- [AFLS01] KIM ALLEMAND, KOMEI FUKUDA, THOMAS M. LIEBLING, AND ERICH STEINER. A polynomial case of unconstrained zero-one quadratic optimization. *Math. Program.*, **91**(1, Ser. A):49–52, 2001.
- [Bal01] EGON BALAS. Projection and lifting in combinatorial optimization. In *Computational combinatorial optimization (Schloß Dagstuhl, 2000)*, Volume **2241** of *Lecture Notes in Comput. Sci.*, pages 26–56. Springer, Berlin, 2001.
- [BB98] TAMAS BADICS AND ENDRE BOROS. Minimization of half-products. *Math. Oper. Res.*, **23**(3):649–660, 1998.
- [BP83] EGON BALAS AND WILLIAM R. PULLEYBLANK. The perfectly matchable subgraph polytope of a bipartite graph. *Networks*, **13**(4):495–516, 1983.
- [BP89] EGON BALAS AND WILLIAM R. PULLEYBLANK. The perfectly matchable subgraph polytope of an arbitrary graph. *Combinatorica*, **9**(4):321–337, 1989.
- [CGLL03] DON COPPERSMITH, OKTAY GÜNLÜK, JON LEE, AND JANNY LEUNG. A polytope for a product of real linear functions in 0/1 variables, 2003. Manuscript, November 2003.
- [CK05] JINLIANG CHENG AND WIESLAW KUBIAK. A half-product based approximation scheme for agreeably weighted completion time variance. *European J. Oper. Res.*, **162**(1):45–54, 2005.
- [ÇKM06] ERANDA ÇELA, BETTINA KLINZ, AND CHRISTOPHE MEYER. Polynomially solvable cases of the constant rank unconstrained quadratic 0-1 programming problem. *J. Comb. Optim.*, **12**(3):187–215, 2006.
- [CLL99] DON COPPERSMITH, JON LEE, AND JANNY LEUNG. A polytope for a product of real linear functions in 0/1 variables, 1999. IBM Research Report RC21568, September 1999.
- [DL97] MICHEL DEZA AND MONIQUE LAURENT. *Geometry of cuts and metrics*. Springer-Verlag, Berlin, 1997.
- [DS90] CATERINA DE SIMONE. The cut polytope and the Boolean quadric polytope. *Discrete Math.*, **79**(1):71–75, 1989/90.
- [FFL05] JEAN-ALBERT FERREZ, KOMEI FUKUDA, AND THOMAS M. LIEBLING. Solving the fixed rank convex quadratic maximization in binary variables by a parallel zonotope construction algorithm. *European J. Oper. Res.*, **166**(1):35–50, 2005.
- [GLS81] MARTIN GRÖTSCHEL, LÁSZLÓ LOVÁSZ, AND ALEXANDER SCHRIJVER. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, **1**(2):169–197, 1981.
- [GLS84] MARTIN GRÖTSCHEL, LÁSZLÓ LOVÁSZ, AND ALEXANDER SCHRIJVER. Corrigendum to our paper: “The ellipsoid method and its consequences in combinatorial optimization”. *Combinatorica*, **4**(4):291–295, 1984.

- [GLS93] MARTIN GRÖTSCHEL, LÁSZLÓ LOVÁSZ, AND ALEXANDER SCHRIJVER. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, Berlin, second edition, 1993.
- [HHPR02] PETER L. HAMMER, PIERRE HANSEN, PANOS M. PARDALOS, AND DAVID J. RADER, JR. Maximizing the product of two linear functions in 0-1 variables. *Optimization*, **51**(3):511–537, 2002.
- [HK56] ALAN J. HOFFMAN AND JOSEPH B. KRUSKAL. Integral boundary points of convex polyhedra. In *Linear inequalities and related systems*, pages 223–246. Princeton University Press, Princeton, N. J., 1956. Annals of Mathematics Studies, no. **38**.
- [JKKW05] ADAM JANIAK, MIKHAIL Y. KOVALYOV, WIESŁAW KUBIAK, AND FRANK WERNER. Positive half-products and scheduling with controllable processing times. *European J. Oper. Res.*, **165**(2):416–422, 2005.
- [Kub05] WIESŁAW KUBIAK. Minimization of ordered, symmetric half-products. *Discrete Appl. Math.*, **146**(3):287–300, 2005.
- [Lee07] JON LEE. In situ column generation for a cutting-stock problem. *Computers and Operations Research*, **34**(8):2345–2358, 2007.
- [NW88] GEORGE L. NEMHAUSER AND LAURENCE A. WOLSEY. *Integer and combinatorial optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., New York, 1988. A Wiley-Interscience Publication.
- [Pad89] MANFRED W. PADBERG. The Boolean quadric polytope: Some characteristics, facets and relatives. *Math. Programming, Ser. B*, **45**(1):139–172, 1989.
- [Pit91] ITAMAR PITOWSKY. Correlation polytopes: their geometry and complexity. *Math. Programming*, **50**(3, (Ser. A)):395–414, 1991.

PART VIII:

COMPLEXITY

ON THE COMPLEXITY OF NONLINEAR MIXED-INTEGER OPTIMIZATION

MATTHIAS KÖPPE*

Abstract. This is a survey on the computational complexity of nonlinear mixed-integer optimization. It highlights a selection of important topics, ranging from incomputability results that arise from number theory and logic, to recently obtained fully polynomial time approximation schemes in fixed dimension, and to strongly polynomial-time algorithms for special cases.

1. Introduction. In this survey we study the computational complexity of nonlinear mixed-integer optimization problems, i.e., models of the form

$$\begin{aligned} \max/\min \quad & f(x_1, \dots, x_n) \\ \text{s.t.} \quad & g_1(x_1, \dots, x_n) \leq 0 \\ & \vdots \\ & g_m(x_1, \dots, x_n) \leq 0 \\ & \mathbf{x} \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}, \end{aligned} \tag{1.1}$$

where $n_1 + n_2 = n$ and $f, g_1, \dots, g_m: \mathbb{R}^n \rightarrow \mathbb{R}$ are arbitrary nonlinear functions.

This is a very rich topic. From the very beginning, questions such as how to present the problem to an algorithm, and, in view of possible irrational outcomes, what it actually means to solve the problem need to be answered. Fundamental intractability results from number theory and logic on the one hand and from continuous optimization on the other hand come into play. The spectrum of theorems that we present ranges from incomputability results, to hardness and inapproximability theorems, to classes that have efficient approximation schemes, or even polynomial-time or strongly polynomial-time algorithms.

We restrict our attention to deterministic algorithms in the usual bit complexity (Turing) model of computation. Some of the material in the present survey also appears in [31]. For an excellent recent survey focusing on other aspects of the complexity of nonlinear optimization, including the performance of oracle-based models and combinatorial settings such as nonlinear network flows, we refer to Hochbaum [34]. We also do not cover the recent developments by Onn et al. [11–13, 21–23, 32, 44, 45] in the context of discrete convex optimization, for which we refer to the monograph [53]. Other excellent sources are [16] and [55].

*Dept. of Mathematics, University of California, Davis, One Shields Avenue, Davis, CA 95616, USA (mkoeppe@math.ucdavis.edu).

2. Preliminaries.

2.1. Presentation of the problem. We restrict ourselves to a model where the problem is presented explicitly. In most of this survey, the functions f and g_i will be polynomial functions presented in a sparse encoding, where all coefficients are rational (or integer) and encoded in the binary scheme. It is useful to assume that the exponents of monomials are given in the unary encoding scheme; otherwise already in very simple cases the results of function evaluations will have an encoding length that is exponential in the input size.

In an alternative model, the functions are presented by oracles, such as comparison oracles or evaluation oracles. This model permits to handle more general functions (not just polynomials), and on the other hand it is very useful to obtain hardness results.

2.2. Encoding issues for solutions. When we want to study the computational complexity of these optimization problems, we first need to discuss how to encode the input (the data of the optimization problem) and the output (an optimal solution if it exists). In the context of *linear* mixed-integer optimization, this is straightforward: Seldom are we concerned with irrational objective functions or constraints; when we restrict the input to be rational as is usual, then also optimal solutions will be rational.

This is no longer true even in the easiest cases of nonlinear optimization, as can be seen on the following quadratically constrained problem in one continuous variable:

$$\max \quad f(x) = x^4 \quad \text{s.t.} \quad x^2 \leq 2.$$

Here the unique optimal solution is irrational ($x^* = \sqrt{2}$, with $f(x^*) = 4$), and so it does not have a finite binary encoding. We ignore here the possibilities of using a model of computation and complexity over the real numbers, such as the celebrated Blum–Shub–Smale model [14]. In the familiar Turing model of computation, we need to resort to approximations.

In the example above it is clear that for every $\epsilon > 0$, there exists a rational x that is a *feasible* solution for the problem and satisfies $|x - x^*| < \epsilon$ (proximity to the optimal solution) or $|f(x) - f(x^*)| < \epsilon$ (proximity to the optimal value). However, in general we cannot expect to find approximations by feasible solutions, as the following example shows.

$$\max \quad f(x) = x \quad \text{s.t.} \quad x^3 - 2x = 0.$$

(Again, the optimal solution is $x = \sqrt{2}$, but the closest rational feasible solution is $x = 0$.) Thus, in the general situation, we will have to use the following notion of approximation:

DEFINITION 2.1. *An algorithm \mathcal{A} is said to efficiently approximate an optimization problem if, for every value of the input parameter $\epsilon > 0$, it returns a rational vector \mathbf{x} (not necessarily feasible) with $\|\mathbf{x} - \mathbf{x}^*\| \leq \epsilon$,*

where \mathbf{x}^* is an optimal solution, and the running time of \mathcal{A} is polynomial in the input encoding of the instance and in $\log 1/\epsilon$.

2.3. Approximation algorithms and schemes. The polynomial dependence of the running time in $\log 1/\epsilon$, as defined above, is a very strong requirement. For many problems, efficient approximation algorithms of this type do not exist, unless $P = NP$. The following, weaker notions of approximation are useful; here it is common to ask for the approximations to be *feasible solutions*, though.

DEFINITION 2.2.

- (a) An algorithm \mathcal{A} is an ϵ -approximation algorithm for a maximization problem with optimal cost f_{\max} , if for each instance of the problem of encoding length n , \mathcal{A} runs in polynomial time in n and returns a feasible solution with cost $f_{\mathcal{A}}$, such that

$$f_{\mathcal{A}} \geq (1 - \epsilon) \cdot f_{\max}. \quad (2.1)$$

- (b) A family of algorithms \mathcal{A}_ϵ is a polynomial time approximation scheme (PTAS) if for every error parameter $\epsilon > 0$, \mathcal{A}_ϵ is an ϵ -approximation algorithm and its running time is polynomial in the size of the instance for every fixed ϵ .
- (c) A family $\{\mathcal{A}_\epsilon\}_\epsilon$ of ϵ -approximation algorithms is a fully polynomial time approximation scheme (FPTAS) if the running time of \mathcal{A}_ϵ is polynomial in the encoding size of the instance and $1/\epsilon$.

These notions of approximation are the usual ones in the domain of combinatorial optimization. It is clear that they are only useful when the function f (or at least the maximal value f_{\max}) are non-negative. For polynomial or general nonlinear optimization problems, various authors [9, 17, 65] have proposed to use a different notion of approximation, where we compare the approximation error to the *range* of the objective function on the feasible region,

$$|f_{\mathcal{A}} - f_{\max}| \leq \epsilon |f_{\max} - f_{\min}|. \quad (2.2)$$

(Here f_{\min} denotes the minimal value of the function on the feasible region.) It enables us to study objective functions that are not restricted to be non-negative on the feasible region. In addition, this notion of approximation is invariant under shifting of the objective function by a constant, and under exchanging minimization and maximization. On the other hand, it is not useful for optimization problems that have an infinite range. We remark that, when the objective function can take negative values on the feasible region, (2.2) is weaker than (2.1). We will call approximation algorithms and schemes with respect to this notion of approximation *weak*. This terminology, however, is not consistent in the literature; [16], for instance, uses the notion (2.2) without an additional attribute and instead reserves the word *weak* for approximation algorithms and schemes that give a guarantee on the absolute error:

$$|f_{\mathcal{A}} - f_{\max}| \leq \epsilon. \quad (2.3)$$

3. Incomputability. Before we can even discuss the computational complexity of nonlinear mixed-integer optimization, we need to be aware of fundamental incomputability results that preclude the existence of *any* algorithm to solve general integer polynomial optimization problems.

Hilbert's tenth problem asked for an algorithm to decide whether a given multivariate polynomial $p(x_1, \dots, x_n)$ has an integer root, i.e., whether the Diophantine equation

$$p(x_1, \dots, x_n) = 0, \quad x_1, \dots, x_n \in \mathbb{Z} \quad (3.1)$$

is solvable. It was answered in the negative by Matiyasevich [48], based on earlier work by Davis, Putnam, and Robinson; see also [49]. A short self-contained proof, using register machines, is presented in [39].

THEOREM 3.1.

- (i) *There does not exist an algorithm that, given polynomials p_1, \dots, p_m , decides whether the system $p_i(x_1, \dots, x_n) = 0$, $i = 1, \dots, m$, has a solution over the integers.*
- (ii) *There does not exist an algorithm that, given a polynomial p , decides whether $p(x_1, \dots, x_n) = 0$ has a solution over the integers.*
- (iii) *There does not exist an algorithm that, given a polynomial p , decides whether $p(x_1, \dots, x_n) = 0$ has a solution over the non-negative integers $\mathbb{Z}_+ = \{0, 1, 2, \dots\}$.*
- (iv) *There does not exist an algorithm that, given a polynomial p , decides whether $p(x_1, \dots, x_n) = 0$ has a solution over the natural numbers $\mathbb{N} = \{1, 2, \dots\}$.*

These three variants of the statement are easily seen to be equivalent. The solvability of the system $p_i(x_1, \dots, x_n) = 0$, $i = 1, \dots, m$, is equivalent to the solvability of $\sum_{i=1}^m p_i^2(x_1, \dots, x_n) = 0$. Also, if $(x_1, \dots, x_n) \in \mathbb{Z}^n$ is a solution of $p(x_1, \dots, x_n) = 0$ over the integers, then by splitting variables into their positive and negative parts, $y_i = \max\{0, x_i\}$ and $z_i = \max\{0, -x_i\}$, clearly $(y_1, z_1; \dots; y_n, z_n)$ is a non-negative integer solution of the polynomial equation $q(y_1, z_1; \dots; y_n, z_n) := p(y_1 - z_1, \dots, y_n - z_n) = 0$. (A construction with only one extra variable is also possible: Use the non-negative variables $w = \max\{|x_i| : x_i < 0\}$ and $y_i := x_i + w$.) In the other direction, using Lagrange's four-square theorem, any non-negative integer x can be represented as the sum $a^2 + b^2 + c^2 + d^2$ with integers a, b, c, d . Thus, if $(x_1, \dots, x_n) \in \mathbb{Z}_+^n$ is a solution over the non-negative integers, then there exists a solution $(a_1, b_1, c_1, d_1; \dots; a_n, b_n, c_n, d_n)$ of the polynomial equation $r(a_1, b_1, c_1, d_1; \dots; a_n, b_n, c_n, d_n) := p(a_1^2 + b_1^2 + c_1^2 + d_1^2, \dots, a_n^2 + b_n^2 + c_n^2 + d_n^2) = 0$. The equivalence of the statement with non-negative integers and the one with the natural numbers follows from a simple change of variables.

Sharper statements of the above incomputability result can be found in [38]. All incomputability statements appeal to the classic result by Turing [64] on the existence of recursively enumerable (or listable) sets of natu-

ral numbers that are not recursive, such as the halting problem of universal Turing machines.

THEOREM 3.2. *For the following universal pairs (ν, δ)*

$$(58, 4), \dots, (38, 8), \dots, (21, 96), \dots, (14, 2.0 \times 10^5), \dots, (9, 1.638 \times 10^{45}),$$

there exists a universal polynomial $U(x; z, u, y; a_1, \dots, a_\nu)$ of degree δ in $4 + \nu$ variables, i.e., for every recursively enumerable (listable) set X there exist natural numbers z, u, y , such that

$$x \in X \iff \exists a_1, \dots, a_\nu \in \mathbb{N} : U(x; z, u, y; a_1, \dots, a_\nu) = 0.$$

Jones explicitly constructs these universal polynomials, using and extending techniques by Matiyasevich. Jones also constructs an explicit system of quadratic equations in $4 + 58$ variables that is universal in the same sense. The reduction of the degree, down to 2, works at the expense of introducing additional variables; this technique goes back to Skolem [62].

In the following, we highlight some of the consequences of these results. Let U be a universal polynomial corresponding to a universal pair (ν, δ) , and let X be a recursively enumerable set that is not recursive, i.e., there does not exist any algorithm (Turing machine) to decide whether a given x is in X . By the above theorem, there exist natural numbers z, u, y such that $x \in X$ holds if and only if the polynomial equation $U(x; z, u, y; a_1, \dots, a_\nu) = 0$ has a solution in natural numbers a_1, \dots, a_ν (note that x and z, u, y are fixed parameters here). This implies:

THEOREM 3.3.

- (i) *Let (ν, δ) be any of the universal pairs listed above. Then there does not exist any algorithm that, given a polynomial p of degree at most δ in ν variables, decides whether $p(x_1, \dots, x_n) = 0$ has a solution over the non-negative integers.*
- (ii) *In particular, there does not exist any algorithm that, given a polynomial p in at most 9 variables, decides whether $p(x_1, \dots, x_n) = 0$ has a solution over the non-negative integers.*
- (iii) *There also does not exist any algorithm that, given a polynomial p in at most 36 variables, decides whether $p(x_1, \dots, x_n) = 0$ has a solution over the integers.*
- (iv) *There does not exist any algorithm that, given a polynomial p of degree at most 4, decides whether $p(x_1, \dots, x_n) = 0$ has a solution over the non-negative integers (or over the integers).*
- (v) *There does not exist any algorithm that, given a system of quadratic equations in at most 58 variables, decides whether it has a solution of the non-negative integers.*
- (vi) *There does not exist any algorithm that, given a system of quadratic equations in at most 232 variables, decides whether it has a solution of the integers.*

We remark that the bounds of $4 \times 9 = 36$ and $4 \times 58 = 232$ are most probably not sharp; they are obtained by a straightforward application of the reduction using Lagrange's theorem.

For integer polynomial optimization, this has the following fundamental consequences. First of all, Theorem 3.3 can be understood as a statement on the feasibility problem of an integer polynomial optimization problem. Thus, the feasibility of an integer polynomial optimization problem with a single polynomial constraint in 9 non-negative integer variables or 36 free integer variables is undecidable, etc.

If we wish to restrict our attention to *feasible* optimization problems, we can consider the problem of minimizing $p^2(x_1, \dots, x_n)$ over the integers or non-negative integers and conclude that unconstrained polynomial optimization in 9 non-negative integer or 36 free integer variables is undecidable. We can also follow Jeroslow [37] and associate with an arbitrary polynomial p in n variables the optimization problem

$$\begin{aligned} \min \quad & u \\ \text{s.t.} \quad & (1-u) \cdot p(x_1, \dots, x_n) = 0, \\ & u \in \mathbb{Z}_+, \quad \mathbf{x} \in \mathbb{Z}_+^n. \end{aligned}$$

This optimization problem is always feasible and has the optimal solution value 0 if and only if $p(x_1, \dots, x_n) = 0$ is solvable, and 1 otherwise. Thus, optimizing *linear forms* over one polynomial constraint in 10 non-negative integer variables is incomputable, and similar statements can be derived from the other universal pairs above. Jeroslow [37] used the above program and a degree reduction (by introducing additional variables) to prove the following.

THEOREM 3.4. *The problem of minimizing a linear form over quadratic inequality constraints in integer variables is not computable; this still holds true for the subclass of problems that are feasible, and where the minimum value is either 0 or 1.*

This statement can be strengthened by giving a bound on the number of integer variables.

4. Hardness and inapproximability. All incomputability results, of course, no longer apply when finite bounds for all variables are known; in this case, a trivial enumeration approach gives a finite algorithm. This is immediately the case when finite bounds for all variables are given in the problem formulation, such as for 0-1 integer problems.

For other problem classes, even though finite bounds are not given, it is possible to compute such bounds that either hold for all feasible solutions or for an optimal solution (if it exists). This is well-known for the case of linear constraints, where the usual encoding length estimates of basic solutions [26] are available. As we explain in section 6.2 below, such finite bounds can also be computed for convex and quasi-convex integer optimization problems.

In other cases, algorithms to decide feasibility exist even though no finite bounds for the variables are known. An example is the case of single Diophantine equations of degree 2, which are decidable using an algorithm by Siegel [61]. We discuss the complexity of this case below.

Within any such computable subclass, we can ask the question of the complexity. Below we discuss hardness results that come from the number theoretic side of the problem (section 4.1) and those that come from the continuous optimization side (section 4.2).

4.1. Hardness results from quadratic Diophantine equations in fixed dimension. The computational complexity of single quadratic Diophantine equations in 2 variables is already very interesting and rich in structure; we refer to the excellent paper by Lagarias [43]. Below we discuss some of these aspects and their implications on optimization.

Testing primality of a number N is equivalent to deciding feasibility of the equation

$$(x + 2)(y + 2) = N \quad (4.1)$$

over the non-negative integers. Recently, Agrawal, Kayal, and Saxena [2] showed that primality can be tested in polynomial time. However, the complexity status of *finding* factors of a composite number, i.e., finding a solution (x, y) of (4.1), is still unclear.

The class also contains subclasses of NP-complete feasibility problems, such as the problem of deciding for given $\alpha, \beta, \gamma \in \mathbb{N}$ whether there exist $x_1, x_2 \in \mathbb{Z}_+$ with $\alpha x_1^2 + \beta x_2 = \gamma$ [47]. On the other hand, the problem of deciding for given $a, c \in \mathbb{N}$ whether there exist $x_1, x_2 \in \mathbb{Z}$ with $ax_1 x_2 + x_2 = c$, lies in $\text{NP} \setminus \text{coNP}$ unless $\text{NP} = \text{coNP}$ [1].

The feasibility problem of the general class of quadratic Diophantine equations in two (non-negative) variables was shown by Lagarias [43] to be in NP. This is not straightforward because minimal solutions can have an encoding size that is exponential in the input size. This can be seen in the case of the so-called *anti-Pellian equation* $x^2 - dy^2 = -1$. Here Lagarias [42] proved that for all $d = 5^{2n+1}$, there exists a solution, and the solution with minimal binary encoding length has an encoding length of $\Omega(5^n)$ (while the input is of encoding length $\Theta(n)$). (We remark that the special case of the anti-Pellian equation is in coNP, as well.)

Related hardness results include the problem of quadratic congruences with a bound, i.e., deciding for given $a, b, c \in \mathbb{N}$ whether there exists a positive integer $x < c$ with $x^2 \equiv a \pmod{b}$; this is the NP-complete problem AN1 in [25].

From these results, we immediately get the following consequences on optimization.

THEOREM 4.1.

- (i) *The feasibility problem of quadratically constrained problems in $n = 2$ integer variables is NP-complete.*

- (ii) *The problems of computing a feasible (or optimal) solution of quadratically constrained problems in $n = 2$ integer variables is not polynomial-time solvable (because the output may require exponential space).*
- (iii) *The feasibility problem of quadratically constrained problems in $n > 2$ integer variables is NP-hard (but it is unknown whether it is in NP).*
- (iv) *The problem of minimizing a degree-4 polynomial over the lattice points of a convex polygon (dimension $n = 2$) is NP-hard.*
- (v) *The problem of finding the minimal value of a degree-4 polynomial over \mathbb{Z}_+^2 is NP-hard; writing down an optimal solution cannot be done in polynomial time.*

However, the complexity of minimizing a quadratic form over the integer points in polyhedra of fixed dimension is unclear, even in dimension $n = 2$. Consider the integer convex minimization problem

$$\begin{aligned} \min \quad & \alpha x_1^2 + \beta x_2, \\ \text{s.t.} \quad & x_1, x_2 \in \mathbb{Z}_+ \end{aligned}$$

for $\alpha, \beta \in \mathbb{N}$. Here an optimal solution can be obtained efficiently, as we explain in section 6.2; in fact, clearly $x_1 = x_2 = 0$ is the unique optimal solution. On the other hand, the problem whether there exists a point (x_1, x_2) of a prescribed objective value $\gamma = \alpha x_1^2 + \beta x_2$ is NP-complete (see above). For indefinite quadratic forms, even in dimension 2, nothing seems to be known.

In varying dimension, the convex quadratic maximization case, i.e., maximizing positive definite quadratic forms is an NP-hard problem. This is even true in very restricted settings such as the problem to maximize $\sum_i (\mathbf{w}_i^\top \mathbf{x})^2$ over $\mathbf{x} \in \{0, 1\}^n$ [53].

4.2. Inapproximability of nonlinear optimization in varying dimension. Even in the pure continuous case, nonlinear optimization is known to be hard. Bellare and Rogaway [9, 10] proved the following inapproximability results using the theory of interactive proof systems.

THEOREM 4.2. *Assume that P \neq NP.*

- (i) *For any $\epsilon < \frac{1}{3}$, there does not exist a polynomial-time weak ϵ -approximation algorithm for the problem of (continuous) quadratic programming over polytopes.*
- (ii) *There exists a constant $\delta > 0$ such that the problem of polynomial programming over polytopes does not have a polynomial-time weak $(1 - n^{-\delta})$ -approximation algorithm.*

Here the number $1 - n^{-\delta}$ becomes arbitrarily close to 0 for growing n ; note that a weak 0-approximation algorithm is one that gives no guarantee other than returning a feasible solution.

Inapproximability still holds for the special case of minimizing a quadratic form over the cube $[-1, 1]^n$ or over the standard simplex. In the case of the cube, inapproximability of the max-cut problem is used. In

the case of the standard simplex, it follows via the celebrated Motzkin–Straus theorem [51] from the inapproximability of the maximum stable set problem. These are results by Håstad [28]; see also [16].

5. Approximation schemes. For important classes of optimization problems, while exact optimization is hard, good approximations can still be obtained efficiently.

Many such examples are known in combinatorial settings. As an example in continuous optimization, we refer to the problem of maximizing homogeneous polynomial functions of fixed degree over simplices. Here de Klerk et al. [17] proved a weak PTAS.

Below we present a general result for mixed-integer polynomial optimization over polytopes.

5.1. Mixed-integer polynomial optimization in fixed dimension over linear constraints: FPTAS and weak FPTAS. Here we consider the problem

$$\begin{aligned} \max/\min \quad & f(x_1, \dots, x_n) \\ \text{subject to} \quad & Ax \leq b \\ & \mathbf{x} \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}, \end{aligned} \tag{5.1}$$

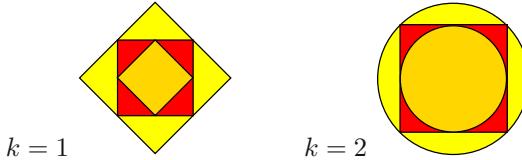
where A is a rational matrix and \mathbf{b} is a rational vector. As we pointed out above (Theorem 4.1), optimizing degree-4 polynomials over problems with two integer variables ($n_1 = 0, n_2 = 2$) is already a hard problem. Thus, even when we fix the dimension, we cannot get a polynomial-time algorithm for solving the optimization problem. The best we can hope for, even when the number of both the continuous and the integer variables is fixed, is an approximation result.

We present here the FPTAS obtained by De Loera et al. [18–20], which uses the “summation method” and the theory of *short rational generating functions* pioneered by Barvinok [6, 7]. We review the methods below; the FPTAS itself appears in Section 5.1. An open question is briefly discussed at the end of this section.

5.1.1. The summation method. The summation method for optimization is the idea to use of elementary relation

$$\max\{s_1, \dots, s_N\} = \lim_{k \rightarrow \infty} \sqrt[k]{s_1^k + \dots + s_N^k}, \tag{5.2}$$

which holds for any finite set $S = \{s_1, \dots, s_N\}$ of non-negative real numbers. This relation can be viewed as an approximation result for ℓ_k -norms. Now if P is a polytope and f is an objective function non-negative on $P \cap \mathbb{Z}^d$, let $\mathbf{x}^1, \dots, \mathbf{x}^N$ denote all the feasible integer solutions in $P \cap \mathbb{Z}^d$ and collect their objective function values $s_i = f(\mathbf{x}^i)$ in a vector $\mathbf{s} \in \mathbb{Q}^N$.

FIG. 1. Approximation properties of ℓ_k -norms.

Then, comparing the unit balls of the ℓ_k -norm and the ℓ_∞ -norm (Figure 1), we get the relation

$$L_k := N^{-1/k} \|\mathbf{s}\|_k \leq \|\mathbf{s}\|_\infty \leq \|\mathbf{s}\|_k =: U_k.$$

These estimates are independent of the function f . (Different estimates that make use of the properties of f , and that are suitable also for the continuous case, can be obtained from the Hölder inequality; see for instance [3].)

Thus, for obtaining a good approximation of the maximum, it suffices to solve a summation problem of the polynomial function $h = f^k$ on $P \cap \mathbb{Z}^d$ for a value of k that is large enough. Indeed, for $k = \lceil (1 + 1/\epsilon) \log N \rceil$, we obtain $U_k - L_k \leq \epsilon f(\mathbf{x}^{\max})$. On the other hand, this choice of k is polynomial in the input size (because $1/\epsilon$ is encoded in unary in the input, and $\log N$ is bounded by a polynomial in the binary encoding size of the polytope P). Hence, when the dimension d is fixed, we can expand the polynomial function f^k as a list of monomials in polynomial time.

5.1.2. Rational generating functions. To solve the summation problem, one uses the technology of short rational generating functions. We explain the theory on a simple, one-dimensional example. Let us consider the set S of integers in the interval $P = [0, \dots, n]$. We associate with S the polynomial $g(P; z) = z^0 + z^1 + \dots + z^{n-1} + z^n$; i.e., every integer $\alpha \in S$ corresponds to a monomial z^α with coefficient 1 in the polynomial $g(P; z)$. This polynomial is called the *generating function* of S (or of P). From the viewpoint of computational complexity, this generating function is of exponential size (in the encoding length of n), just as an explicit list of all the integers $0, 1, \dots, n-1, n$ would be. However, we can observe that $g(P; z)$ is a finite geometric series, so there exists a simple summation formula that expresses it in a much more compact way:

$$g(P; z) = z^0 + z^1 + \dots + z^{n-1} + z^n = \frac{1 - z^{n+1}}{1 - z}. \quad (5.3)$$

The “long” polynomial has a “short” representation as a rational function. The encoding length of this new formula is *linear* in the encoding length of n . On the basis of this idea, we can solve the summation problem. Consider the generating function of the interval $P = [0, 4]$,

$$g(P; z) = z^0 + z^1 + z^2 + z^3 + z^4 = \frac{1}{1-z} - \frac{z^5}{1-z}.$$

We now apply the differential operator $z \frac{d}{dz}$ and obtain

$$\left(z \frac{d}{dz} \right) g(P; z) = 1z^1 + 2z^2 + 3z^3 + 4z^4 = \frac{1}{(1-z)^2} - \frac{-4z^5 + 5z^4}{(1-z)^2}.$$

Applying the same differential operator again, we obtain

$$\begin{aligned} \left(z \frac{d}{dz} \right) \left(z \frac{d}{dz} \right) g(P; z) &= 1z^1 + 4z^2 + 9z^3 + 16z^4 \\ &= \frac{z + z^2}{(1-z)^3} - \frac{25z^5 - 39z^6 + 16z^7}{(1-z)^3}. \end{aligned}$$

We have thus evaluated the monomial function $h(\alpha) = \alpha^2$ for $\alpha = 0, \dots, 4$; the results appear as the coefficients of the respective monomials. Substituting $z = 1$ yields the desired sum

$$\left(z \frac{d}{dz} \right) \left(z \frac{d}{dz} \right) g(P; z) \Big|_{z=1} = 1 + 4 + 9 + 16 = 30.$$

The idea now is to evaluate this sum instead by computing the limit of the rational function for $z \rightarrow 1$,

$$\sum_{\alpha=0}^4 \alpha^2 = \lim_{z \rightarrow 1} \left[\frac{z + z^2}{(1-z)^3} - \frac{25z^5 - 39z^6 + 16z^7}{(1-z)^3} \right];$$

this can be done using residue techniques.

We now present the general definitions and results. Let $P \subseteq \mathbb{R}^d$ be a rational polyhedron. We first define its *generating function* as the *formal Laurent series* $\tilde{g}(P; \mathbf{z}) = \sum_{\alpha \in P \cap \mathbb{Z}^d} \mathbf{z}^\alpha \in \mathbb{Z}[[z_1, \dots, z_d, z_1^{-1}, \dots, z_d^{-1}]]$, i.e., without any consideration of convergence properties. By convergence, one moves to a *rational generating function* $g(P; \mathbf{z}) \in \mathbb{Q}(z_1, \dots, z_d)$.

The following breakthrough result was obtained by Barvinok in 1994.

THEOREM 5.1 (Barvinok [6]). *Let d be fixed. Then there exists a polynomial-time algorithm for computing the generating function $g(P; \mathbf{z})$ of a polyhedron $P \subseteq \mathbb{R}^d$ given by rational inequalities in the form of a rational function*

$$g(P; \mathbf{z}) = \sum_{i \in I} \epsilon_i \frac{\mathbf{z}^{\mathbf{a}_i}}{\prod_{j=1}^d (1 - \mathbf{z}^{\mathbf{b}_{ij}})} \quad (5.4)$$

with $\epsilon_i \in \{\pm 1\}$, $\mathbf{a}_i \in \mathbb{Z}^d$, and $\mathbf{b}_{ij} \in \mathbb{Z}^d$.

5.1.3. Efficient summation using rational generating functions. Below we describe the theorems on the summation method based on short rational generating functions, which appeared in [18–20]. Let $g(P; \mathbf{z})$ be the rational generating function of $P \cap \mathbb{Z}^d$, computed using Barvinok’s algorithm. By symbolically applying differential operators to $g(P; \mathbf{z})$, we can compute a short rational function representation of the Laurent polynomial $g(P, h; \mathbf{z}) = \sum_{\alpha \in P \cap \mathbb{Z}^d} h(\alpha) \mathbf{z}^\alpha$, where each monomial \mathbf{z}^α corresponding to an integer point $\alpha \in P \cap \mathbb{Z}^d$ has a coefficient that is the value $h(\alpha)$. As in the one-dimensional example above, we use the partial differential operators $z_i \frac{\partial}{\partial z_i}$ for $i = 1, \dots, d$ on the short rational generating function. In fixed dimension, the size of the rational function expressions occurring in the symbolic calculation can be bounded polynomially. Thus one obtains the following result.

THEOREM 5.2 (see [19], Lemma 3.1).

- (a) Let $h(x_1, \dots, x_d) = \sum_{\beta} c_\beta \mathbf{x}^\beta \in \mathbb{Q}[x_1, \dots, x_d]$ be a polynomial. Define the differential operator

$$D_h = h \left(z_1 \frac{\partial}{\partial z_1}, \dots, z_d \frac{\partial}{\partial z_d} \right) = \sum_{\beta} c_\beta \left(z_1 \frac{\partial}{\partial z_1} \right)^{\beta_1} \dots \left(z_d \frac{\partial}{\partial z_d} \right)^{\beta_d}.$$

Then D_h maps the generating function $g(P; \mathbf{z}) = \sum_{\alpha \in P \cap \mathbb{Z}^d} \mathbf{z}^\alpha$ to the weighted generating function

$$(D_h g)(\mathbf{z}) = g(P, h; \mathbf{z}) = \sum_{\alpha \in P \cap \mathbb{Z}^d} h(\alpha) \mathbf{z}^\alpha.$$

- (b) Let the dimension d be fixed. Let $g(P; \mathbf{z})$ be the Barvinok representation of the generating function $\sum_{\alpha \in P \cap \mathbb{Z}^d} \mathbf{z}^\alpha$ of $P \cap \mathbb{Z}^d$. Let $h \in \mathbb{Q}[x_1, \dots, x_d]$ be a polynomial, given as a list of monomials with rational coefficients c_β encoded in binary and exponents β encoded in unary. We can compute in polynomial time a Barvinok representation $g(P, h; \mathbf{z})$ for the weighted generating function $\sum_{\alpha \in P \cap \mathbb{Z}^d} h(\alpha) \mathbf{z}^\alpha$.

Thus, we can implement the following algorithm in polynomial time (in fixed dimension).

ALGORITHM 1 (Computation of bounds for the optimal value).

Input: A rational convex polytope $P \subset \mathbb{R}^d$; a polynomial objective function $f \in \mathbb{Q}[x_1, \dots, x_d]$ that is non-negative over $P \cap \mathbb{Z}^d$, given as a list of monomials with rational coefficients c_β encoded in binary and exponents β encoded in unary; an index k , encoded in unary.

Output: A lower bound L_k and an upper bound U_k for the maximal function value f^* of f over $P \cap \mathbb{Z}^d$. The bounds L_k form a nondecreasing, the bounds U_k a nonincreasing sequence of bounds that both reach f^* in a finite number of steps.

1. Compute a short rational function expression for the generating function $g(P; \mathbf{z}) = \sum_{\alpha \in P \cap \mathbb{Z}^d} \mathbf{z}^\alpha$. Using residue techniques, compute $|P \cap \mathbb{Z}^d| = g(P; \mathbf{1})$ from $g(P; \mathbf{z})$.

2. Compute the polynomial f^k from f .
3. From the rational function $g(P; \mathbf{z})$ compute the rational function representation of $g(P, f^k; \mathbf{z})$ of $\sum_{\alpha \in P \cap \mathbb{Z}^d} f^k(\boldsymbol{\alpha}) \mathbf{z}^\alpha$ by Theorem 5.2. Using residue techniques, compute

$$L_k := \left\lceil \sqrt[k]{g(P, f^k; \mathbf{1})/g(P; \mathbf{1})} \right\rceil \quad \text{and} \quad U_k := \left\lfloor \sqrt[k]{g(P, f^k; \mathbf{1})} \right\rfloor.$$

From the discussion of the convergence of the bounds, one then obtains the following result.

THEOREM 5.3 (Fully polynomial-time approximation scheme). *Let the dimension d be fixed. Let $P \subset \mathbb{R}^d$ be a rational convex polytope. Let f be a polynomial with rational coefficients that is non-negative on $P \cap \mathbb{Z}^d$, given as a list of monomials with rational coefficients c_β encoded in binary and exponents β encoded in unary.*

- (i) *Algorithm 1 computes the bounds L_k , U_k in time polynomial in k , the input size of P and f , and the total degree D . The bounds satisfy $U_k - L_k \leq f^* \cdot (\sqrt[k]{|P \cap \mathbb{Z}^d|} - 1)$.*
- (ii) *For $k = (1 + 1/\epsilon) \log(|P \cap \mathbb{Z}^d|)$ (a number bounded by a polynomial in the input size), L_k is a $(1 - \epsilon)$ -approximation to the optimal value f^* and it can be computed in time polynomial in the input size, the total degree D , and $1/\epsilon$. Similarly, U_k gives a $(1 + \epsilon)$ -approximation to f^* .*
- (iii) *With the same complexity, by iterated bisection of P , we can also find a feasible solution $\mathbf{x}_\epsilon \in P \cap \mathbb{Z}^d$ with $|f(\mathbf{x}_\epsilon) - f^*| \leq \epsilon f^*$.*

5.1.4. Extension to the mixed-integer case by discretization.

The mixed-integer case can be handled by *discretization* of the continuous variables. We illustrate on an example that one needs to be careful to pick a sequence of discretizations that actually converges. Consider the mixed-integer linear optimization problem depicted in Figure 2, whose feasible region consists of the point $(\frac{1}{2}, 1)$ and the segment $\{(x, 0) : x \in [0, 1]\}$. The unique optimal solution is $x = \frac{1}{2}$, $z = 1$. Now consider the sequence of grid approximations where $x \in \frac{1}{m} \mathbb{Z}_{\geq 0}$. For even m , the unique optimal solution to the grid approximation is $x = \frac{1}{2}$, $z = 1$. However, for odd m , the unique optimal solution is $x = 0$, $z = 0$. Thus the full sequence of the optimal solutions to the grid approximations does not converge because it has two limit points; see Figure 2.

To handle polynomial objective functions that take arbitrary (positive and negative) values, one can shift the objective function by a large constant. Then, to obtain a strong approximation result, one iteratively reduces the constant by a factor. Altogether we have the following result.

THEOREM 5.4 (Fully polynomial-time approximation schemes). *Let the dimension $n = n_1 + n_2$ be fixed. Let an optimization problem (5.1) of a polynomial function f over the mixed-integer points of a polytope P and an error bound ϵ be given, where*

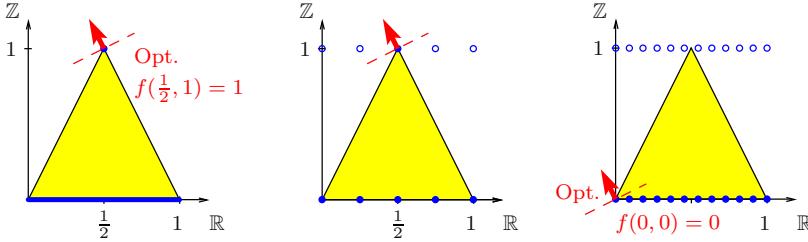


FIG. 2. A mixed-integer linear optimization problem and a sequence of optimal solutions to grid problems with two limit points, for even m and for odd m .

- (I₁) f is given as a list of monomials with rational coefficients c_β encoded in binary and exponents β encoded in unary,
 - (I₂) P is given by rational inequalities in binary encoding,
 - (I₃) the rational number $\frac{1}{\epsilon}$ is given in unary encoding.
- (a) There exists a fully polynomial time approximation scheme (FPTAS) for the maximization problem for all polynomial functions $f(\mathbf{x}, \mathbf{z})$ that are non-negative on the feasible region. That is, there exists a polynomial-time algorithm that, given the above data, computes a feasible solution $(\mathbf{x}_\epsilon, \mathbf{z}_\epsilon) \in P \cap (\mathbb{R}^{n_1} \times \mathbb{Z}^{n_2})$ with

$$|f(\mathbf{x}_\epsilon, \mathbf{z}_\epsilon) - f(\mathbf{x}_{\max}, \mathbf{z}_{\max})| \leq \epsilon f(\mathbf{x}_{\max}, \mathbf{z}_{\max}).$$

- (b) There exists a polynomial-time algorithm that, given the above data, computes a feasible solution $(\mathbf{x}_\epsilon, \mathbf{z}_\epsilon) \in P \cap (\mathbb{R}^{n_1} \times \mathbb{Z}^{n_2})$ with

$$|f(\mathbf{x}_\epsilon, \mathbf{z}_\epsilon) - f(\mathbf{x}_{\max}, \mathbf{z}_{\max})| \leq \epsilon |f(\mathbf{x}_{\max}, \mathbf{z}_{\max}) - f(\mathbf{x}_{\min}, \mathbf{z}_{\min})|.$$

In other words, this is a weak FPTAS.

5.1.5. Open question. Consider the problem (5.1) for a fixed number n_2 of integer variables and a *varying* number n_1 of continuous variables. Of course, even with no integer variables present ($n_2 = 0$), this is NP-hard and inapproximable. On the other hand, if the objective function f is *linear*, the problem can be solved in polynomial time using Lenstra's algorithm. Thus it is interesting to consider the problem for an objective function of restricted nonlinearity, such as

$$f(\mathbf{x}, \mathbf{z}) = g(\mathbf{z}) + \mathbf{c}^\top \mathbf{x},$$

with an arbitrary polynomial function g in the integer variables and a linear form in the continuous variables. The complexity (in particular the existence of approximation algorithms) of this problem is an open question.

6. Polynomial-time algorithms. Here we study important special cases where polynomial-time algorithms can be obtained. We also include cases here where the algorithms efficiently approximate the optimal solution to arbitrary precision, as discussed in section 2.2.

6.1. Fixed dimension: Continuous polynomial optimization.

Here we consider pure continuous polynomial optimization problems of the form

$$\begin{aligned} \min \quad & f(x_1, \dots, x_n) \\ \text{s.t.} \quad & g_1(x_1, \dots, x_n) \leq 0 \\ & \vdots \\ & g_m(x_1, \dots, x_n) \leq 0 \\ & \mathbf{x} \in \mathbb{R}^{n_1}. \end{aligned} \tag{6.1}$$

When the dimension is fixed, this problem can be solved in polynomial time, in the sense that there exists an algorithm that efficiently computes an approximation to an optimal solution. This follows from a much more general theory on the computational complexity of approximating the solutions to general algebraic and semialgebraic formulae over the real numbers by Renegar [60], which we review in the following. The bulk of this theory was developed in [57–59]. Similar results appeared in [29]; see also [8, Chapter 14]). One considers problems associated with logic formulas of the form

$$Q_1 \mathbf{x}^1 \in \mathbb{R}^{n_1} : \dots Q_\omega \mathbf{x}^\omega \in \mathbb{R}^{n_\omega} : P(\mathbf{y}, \mathbf{x}^1, \dots, \mathbf{x}^\omega) \tag{6.2}$$

with quantifiers $Q_i \in \{\exists, \forall\}$, where P is a Boolean combination of polynomial inequalities such as

$$g_i(\mathbf{y}, \mathbf{x}^1, \dots, \mathbf{x}^\omega) \leq 0, \quad i = 1, \dots, m,$$

or using $\geq, <, >$, or $=$ as the relation. Here $\mathbf{y} \in \mathbb{R}^{n_0}$ is a free (i.e., not quantified) variable. Let $d \geq 2$ be an upper bound on the degrees of the polynomials g_i . A vector $\bar{\mathbf{y}} \in \mathbb{R}^{n_0}$ is called a *solution* of this formula if the formula (6.2) becomes a true logic sentence if we set $\mathbf{y} = \bar{\mathbf{y}}$. Let Y denote the set of all solutions. An ϵ -approximate solution is a vector \mathbf{y}_ϵ with $\|\bar{\mathbf{y}} - \mathbf{y}_\epsilon\| < \epsilon$ for some solution $\bar{\mathbf{y}} \in Y$.

The following bound can be proved. When the number ω of “blocks” of quantifiers (i.e., the number of alternations of the quantifiers \exists and \forall) is fixed, then the bound is singly exponential in the dimension.

THEOREM 6.1. *If the formula (6.2) has only integer coefficients of binary encoding size at most ℓ , then every connected component of Y intersects with the ball $\{\|\mathbf{y}\| \leq r\}$, where*

$$\log r = \ell(md)^{2^{O(\omega)} n_0 n_1 \cdots n_k}.$$

This bound is used in the following fundamental result, which gives a general algorithm to compute ϵ -approximate solutions to the formula (6.2).

THEOREM 6.2. *There exists an algorithm that, given numbers $0 < \epsilon < r$ that are integral powers of 2 and a formula (6.2), computes a set $\{\mathbf{y}_i\}_i$ of*

$(md)^{2^{O(\omega)} n_0 n_1 \cdots n_k}$ distinct ϵ -approximate solutions of the formula with the property that for each connected components of $Y \cap \{\|\mathbf{y}\| \leq r\}$ one of the \mathbf{y}_i is within distance ϵ . The algorithm runs in time

$$(md)^{2^{O(\omega)} n_0 n_1 \cdots n_k} (\ell + md + \log \frac{1}{\epsilon} + \log r)^{O(1)}.$$

This can be applied to polynomial optimization problems as follows. Consider the formula

$$\begin{aligned} \forall \mathbf{x} \in \mathbb{R}^{n_1} : g_1(\mathbf{y}) \leq 0 \wedge \cdots \wedge g_m(\mathbf{y}) \leq 0 \\ \wedge [g_1(\mathbf{x}) > 0 \vee \cdots \vee g_m(\mathbf{x}) > 0 \vee f(\mathbf{y}) - f(\mathbf{x}) < 0], \end{aligned} \quad (6.3)$$

this describes that \mathbf{y} is an optimal solution (all other solutions \mathbf{x} are either infeasible or have a higher objective value). Thus optimal solutions can be efficiently approximated using the algorithm of Theorem 6.2.

6.2. Fixed dimension: Convex and quasi-convex integer polynomial minimization. In this section we consider the case of the minimization of convex and quasi-convex polynomials f over the mixed-integer points in convex regions given by convex and quasi-convex polynomial functions g_1, \dots, g_m :

$$\begin{aligned} \min \quad & f(x_1, \dots, x_n) \\ \text{s.t.} \quad & g_1(x_1, \dots, x_n) \leq 0 \\ & \vdots \\ & g_m(x_1, \dots, x_n) \leq 0 \\ & \mathbf{x} \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. \end{aligned} \quad (6.4)$$

Here a function $g: \mathbb{R}^n \rightarrow \mathbb{R}^1$ is called *quasi-convex* if every lower level set $L_\lambda = \{\mathbf{x} \in \mathbb{R}^n : g(\mathbf{x}) \leq \lambda\}$ is a convex set.

The complexity in this setting is fundamentally different from the general (non-convex) case. One important aspect is that bounding results for the coordinates of optimal *integer* solutions exists, which are similar to the ones for continuous solutions in Theorem 6.1 above. For the case of convex functions, these bounding results were obtained by [40, 63]. An improved bound was obtained by [4, 5], which also handles the more general case of quasi-convex polynomials. This bound follows from the efficient theory of quantifier elimination over the real numbers that we referred to in section 6.1.

THEOREM 6.3. *Let $f, g_1, \dots, g_m \in \mathbb{Z}[x_1, \dots, x_n]$ be quasi-convex polynomials of degree at most $d \geq 2$, whose coefficients have a binary encoding length of at most ℓ . Let*

$$F = \left\{ \mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \dots, m \right\}$$

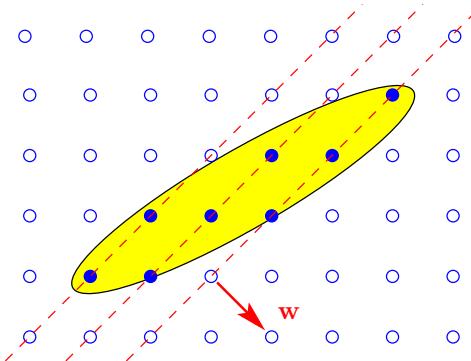


FIG. 3. Branching on hyperplanes corresponding to approximate lattice width directions of the feasible region in a Lenstra-type algorithm.

be the (continuous) feasible region. If the integer minimization problem $\min\{f(\mathbf{x}) : \mathbf{x} \in F \cap \mathbb{Z}^n\}$ is bounded, there exists a radius $R \in \mathbb{Z}_+$ of binary encoding length at most $(md)^{O(n)}\ell$ such that

$$\min\{f(\mathbf{x}) : \mathbf{x} \in F \cap \mathbb{Z}^n\} = \min\{f(\mathbf{x}) : \mathbf{x} \in F \cap \mathbb{Z}^n, \|\mathbf{x}\| \leq R\}.$$

Using this finite bound, a trivial enumeration algorithm can find an optimal solution (but not in polynomial time, not even in fixed dimension). Thus the incomputability result for integer polynomial optimization (Theorem 3.3) does not apply to this case.

The unbounded case can be efficiently detected in the case of quasi-convex polynomials; see [5] and [52], the latter of which also handles the case of “faithfully convex” functions that are not polynomials.

In fixed dimension, the problem of convex integer minimization can be solved efficiently using variants of Lenstra’s algorithm [46] for integer programming. Lenstra-type algorithms are algorithms for solving feasibility problems. We consider a family of feasibility problems associated with the optimization problem,

$$\exists \mathbf{x} \in F_\alpha \cap \mathbb{Z}^n \quad \text{where} \quad F_\alpha = \{\mathbf{x} \in F : f(\mathbf{x}) \leq \alpha\} \quad \text{for } \alpha \in \mathbb{Z}. \quad (6.5)$$

If bounds for $f(\mathbf{x})$ on the feasible regions of polynomial binary encoding size are known, a polynomial time algorithm for this feasibility problem can be used in a binary search to solve the optimization problem in polynomial time. Indeed, when the dimension n is fixed, the bound R given by Theorem 6.3 has a binary encoding size that is bounded polynomially by the input data.

A Lenstra-type algorithm uses branching on hyperplanes (Figure 3) to obtain polynomial time complexity in fixed dimension. Note that only the binary encoding size of the bound R , but not R itself, is bounded polynomially. Thus, multiway branching on the values of a single variable x_i will

create an exponentially large number of subproblems. Instead, a Lenstra-type algorithm computes a primitive lattice vector $\mathbf{w} \in \mathbb{Z}^n$ such that there are only few lattice hyperplanes $\mathbf{w}^\top \mathbf{x} = \gamma$ (with $\gamma \in \mathbb{Z}$) that can have a nonempty intersection with F_α . The *width* of F_α in the direction \mathbf{w} , defined as

$$\max\{\mathbf{w}^\top \mathbf{x} : \mathbf{x} \in F_\alpha\} - \min\{\mathbf{w}^\top \mathbf{x} : \mathbf{x} \in F_\alpha\} \quad (6.6)$$

essentially gives the number of these lattice hyperplanes. A *lattice width direction* is a minimizer of the width among all directions $\mathbf{w} \in \mathbb{Z}^n$, the *lattice width* the corresponding width. Any polynomial bound on the width will yield a polynomial-time algorithm in fixed dimension.

Exact and approximate lattice width directions \mathbf{w} can be constructed using geometry of numbers techniques. We refer to the excellent tutorial [24] and the classic references cited therein. The key to dealing with the feasible region F_α is to apply ellipsoidal rounding. By applying the shallow-cut ellipsoid method (which we describe in more detail below), one finds concentric proportional inscribed and circumscribed ellipsoids that differ by some factor β that only depends on the dimension n . Then any η -approximate lattice width direction for the ellipsoids gives a $\beta\eta$ -approximate lattice width direction for F_α . Lenstra's original algorithm now uses an LLL-reduced basis of the lattice \mathbb{Z}^n with respect to a norm associated with the ellipsoid; the last basis vector then serves as an approximate lattice width direction.

The first algorithm of this kind for convex integer minimization was announced by Khachiyan [40]. In the following we present the variant of Lenstra's algorithm due to Heinz [30], which seems to yield the best complexity bound for the problem published so far. The complexity result is the following.

THEOREM 6.4. *Let $f, g_1, \dots, g_m \in \mathbb{Z}[x_1, \dots, x_n]$ be quasi-convex polynomials of degree at most $d \geq 2$, whose coefficients have a binary encoding length of at most ℓ . There exists an algorithm running in time $m\ell^{O(1)}d^{O(n)}2^{O(n^3)}$ that computes a minimizer $\mathbf{x}^* \in \mathbb{Z}^n$ of the problem (6.4) or reports that no minimizer exists. If the algorithm outputs a minimizer \mathbf{x}^* , its binary encoding size is $\ell d^{O(n)}$.*

We remark that the complexity guarantees can be improved dramatically by combining Heinz' technique with more recent variants of Lenstra's algorithm that rely on the fast computation of shortest vectors [33].

A complexity result of greater generality was presented by Khachiyan and Porkolab [41]. It covers the case of minimization of convex polynomials over the integer points in convex semialgebraic sets given by *arbitrary* (not necessarily quasi-convex) polynomials.

THEOREM 6.5. *Let $Y \subseteq \mathbb{R}^{n_0}$ be a convex set given by*

$$Y = \left\{ \mathbf{y} \in \mathbb{R}^{n_0} : Q_1 \mathbf{x}^1 \in \mathbb{R}^{n_1} : \dots : Q_\omega \mathbf{x}^\omega \in \mathbb{R}^{n_\omega} : P(\mathbf{y}, \mathbf{x}^1, \dots, \mathbf{x}^\omega) \right\}$$

with quantifiers $Q_i \in \{\exists, \forall\}$, where P is a Boolean combination of polynomial inequalities

$$g_i(\mathbf{y}, \mathbf{x}^1, \dots, \mathbf{x}^\omega) \leq 0, \quad i = 1, \dots, m$$

with degrees at most $d \geq 2$ and coefficients of binary encoding size at most ℓ . There exists an algorithm for solving the problem $\min\{y_{n_0} : \mathbf{y} \in Y \cap \mathbb{Z}^{n_0}\}$ in time $\ell^{O(1)}(md)^{O(n_0^4)} \prod_{i=1}^{\omega} O(n_i)$.

When the dimension $n_0 + n_1 + \dots + n_\omega$ is fixed, the algorithm runs in polynomial time. For the case of convex minimization where the feasible region is described by convex polynomials, the complexity bound of Theorem 6.5, however, translates to $\ell^{O(1)}m^{O(n^2)}d^{O(n^4)}$, which is worse than the bound of Theorem 6.4 [30].

In the remainder of this subsection, we describe the ingredients of the variant of Lenstra's algorithm due to Heinz. The algorithm starts out by “rounding” the feasible region, by applying the shallow-cut ellipsoid method to find proportional inscribed and circumscribed ellipsoids. It is well-known [26] that the shallow-cut ellipsoid method only needs an initial circumscribed ellipsoid that is “small enough” (of polynomial binary encoding size – this follows from Theorem 6.3) and an implementation of a *shallow separation oracle*, which we describe below.

For a positive-definite matrix A we denote by $\mathcal{E}(A, \hat{\mathbf{x}})$ the ellipsoid $\{\mathbf{x} \in \mathbb{R}^n : (\mathbf{x} - \hat{\mathbf{x}})^\top A(\mathbf{x} - \hat{\mathbf{x}}) \leq 1\}$.

LEMMA 6.1 (Shallow separation oracle). *Let $g_0, \dots, g_{m+1} \in \mathbb{Z}[\mathbf{x}]$ be quasi-convex polynomials of degree at most d , the binary encoding sizes of whose coefficients are at most r . Let the (continuous) feasible region $F = \{\mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) < 0\}$ be contained in the ellipsoid $\mathcal{E}(A, \hat{\mathbf{x}})$, where A and $\hat{\mathbf{x}}$ have binary encoding size at most ℓ . There exists an algorithm with running time $m(lnr)^{O(1)}d^{O(n)}$ that outputs*

(a) “true” if

$$\mathcal{E}((n+1)^{-3}A, \hat{\mathbf{x}}) \subseteq F \subseteq \mathcal{E}(A, \hat{\mathbf{x}}); \quad (6.7)$$

(b) otherwise, a vector $\mathbf{c} \in \mathbb{Q}^n \setminus \{\mathbf{0}\}$ of binary encoding length $(l+r)(dn)^{O(1)}$ with

$$F \subseteq \mathcal{E}(A, \hat{\mathbf{x}}) \cap \{\mathbf{x} \in \mathbb{R}^n : \mathbf{c}^\top (\mathbf{x} - \hat{\mathbf{x}}) \leq \frac{1}{n+1}(\mathbf{c}^\top A \mathbf{c})^{1/2}\}. \quad (6.8)$$

Proof. We give a simplified sketch of the proof, without hard complexity estimates. By applying an affine transformation to $F \subseteq \mathcal{E}(A, \hat{\mathbf{x}})$, we can assume that F is contained in the unit ball $\mathcal{E}(I, \mathbf{0})$. Let us denote as usual by $\mathbf{e}_1, \dots, \mathbf{e}_n$ the unit vectors and by $\mathbf{e}_{n+1}, \dots, \mathbf{e}_{2n}$ their negatives. The algorithm first constructs numbers $\lambda_{i1}, \dots, \lambda_{id} > 0$ with

$$\frac{1}{n + \frac{3}{2}} < \lambda_{i1} < \dots < \lambda_{id} < \frac{1}{n + 1} \quad (6.9)$$

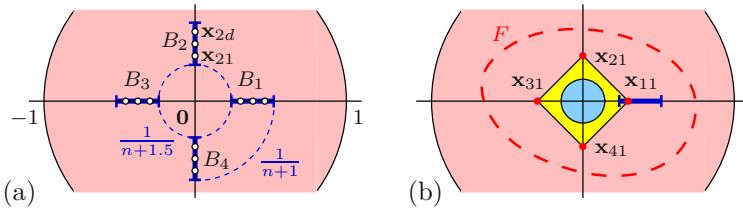


FIG. 4. The implementation of the shallow separation oracle. (a) Test points \mathbf{x}_{ij} in the circumscribed ball $E(1, \mathbf{0})$. (b) Case I: All test points \mathbf{x}_{i1} are (continuously) feasible; so their convex hull (a cross-polytope) and its inscribed ball $E((n+1)^{-3}, \mathbf{0})$ are contained in the (continuous) feasible region F .

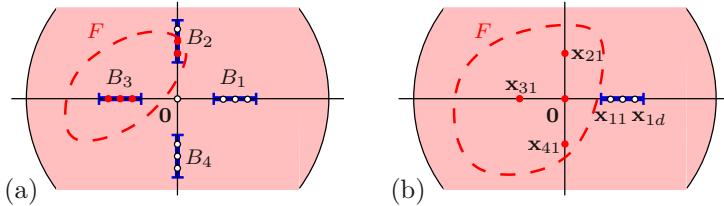


FIG. 5. The implementation of the shallow separation oracle. (a) Case II: The center $\mathbf{0}$ violates a polynomial inequality $g_0(\mathbf{x}) < 0$ (say). Due to convexity, for all $i = 1, \dots, n$, one set of each pair $B_i \cap F$ and $B_{n+i} \cap F$ must be empty. (b) Case III: A test point \mathbf{x}_{k1} is infeasible, as it violates an inequality $g_0(\mathbf{x}) < 0$ (say). However, the center $\mathbf{0}$ is feasible at least for this inequality.

and the corresponding point sets $B_i = \{\mathbf{x}_{ij} := \lambda_{ij}\mathbf{e}_i : j = 1, \dots, d\}$; see Figure 4(a). The choice of the bounds (6.9) for λ_{ij} will ensure that we either find a large enough inscribed ball for (a) or a deep enough cut for (b). Then the algorithm determines the (continuous) feasibility of the center $\mathbf{0}$ and the $2n$ innermost points $\mathbf{x}_{i,1}$.

Case I. If $\mathbf{x}_{i,1} \in F$ for $i = 1, \dots, 2n$, then the cross-polytope $\text{conv}\{\mathbf{x}_{i,1} : i = 1, \dots, 2n\}$ is contained in F ; see Figure 4(b). An easy calculation shows that the ball $E((n+1)^{-3}, \mathbf{0})$ is contained in the cross-polytope and thus in F ; see Figure 4. Hence the condition in (a) is satisfied and the algorithm outputs “true”.

Case II. We now discuss the case when the center $\mathbf{0}$ violates a polynomial inequality $g_0(\mathbf{x}) < 0$ (say). Let $F_0 = \{\mathbf{x} \in \mathbb{R}^n : g_0(\mathbf{x}) < 0\} \supseteq F$. Due to convexity of F_0 , for all $i = 1, \dots, n$, one set of each pair $B_i \cap F_0$ and $B_{n+i} \cap F_0$ must be empty; see Figure 5(a). Without loss of generality, let us assume $B_{n+i} \cap F_0 = \emptyset$ for all i . We can determine whether a n -variate polynomial function of known maximum degree d is constant by evaluating it on $(d+1)^n$ suitable points (this is a consequence of the Fundamental Theorem of Algebra). For our case of quasi-convex polynomials, this can be improved; indeed, it suffices to test whether the gradient ∇g_0 vanishes

on the nd points in the set $B_1 \cup \dots \cup B_n$. If it does, we know that g_0 is constant, thus $F = \emptyset$, and so can we return an arbitrary vector \mathbf{c} . Otherwise, there is a point $\mathbf{x}_{ij} \in B_i$ with $\mathbf{c} := \nabla g_0(\mathbf{x}_{ij}) \neq \mathbf{0}$; we return this vector as the desired normal vector of a shallow cut. Due to the choice of λ_{ij} as a number smaller than $\frac{1}{n+1}$, the cut is deep enough into the ellipsoid $\mathcal{E}(A, \hat{\mathbf{x}})$, so that (6.8) holds.

Case III. The remaining case to discuss is when $\mathbf{0} \in F$ but there exists a $k \in \{1, \dots, 2n\}$ with $\mathbf{x}_{k,1} \notin F$. Without loss of generality, let $k = 1$, and let $\mathbf{x}_{1,1}$ violate the polynomial inequality $g_0(\mathbf{x}) < 0$, i.e., $g_0(\mathbf{x}_{1,1}) \geq 0$; see Figure 5(b). We consider the univariate polynomial $\phi(\lambda) = g_0(\lambda \mathbf{e}_i)$. We have $\phi(0) = g_0(\mathbf{0}) < 0$ and $\phi(\lambda_{1,1}) \geq 0$, so ϕ is not constant. Because ϕ has degree at most d , its derivative ϕ' has degree at most $d-1$, so ϕ' has at most $d-1$ roots. Thus, for at least one of the d different values $\lambda_{1,1}, \dots, \lambda_{1,d}$, say $\lambda_{1,j}$, we must have $\phi'(\lambda_{1,j}) \neq 0$. This implies that $\mathbf{c} := \nabla g_0(\mathbf{x}_{1,j}) \neq \mathbf{0}$. By convexity, we have $\mathbf{x}_{1,j} \notin F$, so we can use \mathbf{c} as the normal vector of a shallow cut. \square

By using this oracle in the shallow-cut ellipsoid method, one obtains the following result.

COROLLARY 6.1. *Let $g_0, \dots, g_m \in \mathbb{Z}[\mathbf{x}]$ be quasi-convex polynomials of degree at most $d \geq 2$. Let the (continuous) feasible region $F = \{ \mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \leq 0 \}$ be contained in the ellipsoid $\mathcal{E}(A_0, \mathbf{0})$, given by the positive-definite matrix $A_0 \in \mathbb{Q}^{n \times n}$. Let $\epsilon \in \mathbb{Q}_{>0}$ be given. Let the entries of A_0 and the coefficients of all monomials of g_0, \dots, g_m have binary encoding size at most ℓ .*

There exists an algorithm with running time $m(\ell n)^{O(1)} d^{O(n)}$ that computes a positive-definite matrix $A \in \mathbb{Q}^{n \times n}$ and a point $\hat{\mathbf{x}} \in \mathbb{Q}^n$ with

- (a) *either $\mathcal{E}((n+1)^{-3} A, \hat{\mathbf{x}}) \subseteq F \subseteq \mathcal{E}(A, \hat{\mathbf{x}})$*
- (b) *or $F \subseteq \mathcal{E}(A, \hat{\mathbf{x}})$ and $\text{vol } \mathcal{E}(A, \hat{\mathbf{x}}) < \epsilon$.*

Finally, there is a lower bound for the volume of a continuous feasible region F that can contain an integer point.

LEMMA 6.2. *Under the assumptions of Corollary 6.1, if $F \cap \mathbb{Z}^n \neq \emptyset$, there exists an $\epsilon \in \mathbb{Q}_{>0}$ of binary encoding size $\ell(dn)^{O(1)}$ with $\text{vol } F > \epsilon$.*

On the basis of these results, one obtains a Lenstra-type algorithm for the decision version of the convex integer minimization problem with the desired complexity. By applying binary search, the optimization problem can be solved, which provides a proof of Theorem 6.4.

6.3. Fixed dimension: Convex integer maximization. Maximizing a convex function over the integer points in a polytope in fixed dimension can be done in polynomial time. To see this, note that the optimal value is taken on at a vertex of the convex hull of all feasible integer points. But when the dimension is fixed, there is only a polynomial number of vertices, as Cook et al. [15] showed.

THEOREM 6.6. *Let $P = \{ \mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b} \}$ be a rational polyhedron with $A \in \mathbb{Q}^{m \times n}$ and let ϕ be the largest binary encoding size of any of the*

rows of the system $Ax \leq b$. Let $P^I = \text{conv}(P \cap \mathbb{Z}^n)$ be the integer hull of P . Then the number of vertices of P^I is at most $2m^n(6n^2\phi)^{n-1}$.

Moreover, Hartmann [27] gave an algorithm for enumerating all the vertices, which runs in polynomial time in fixed dimension.

By using Hartmann's algorithm, we can therefore compute all the vertices of the integer hull P^I , evaluate the convex objective function on each of them and pick the best. This simple method already provides a polynomial-time algorithm.

7. Strongly polynomial-time algorithms: Submodular function minimization. In important specially structured cases, even strongly polynomial-time algorithms are available. The probably most well-known case is that of submodular function minimization. We briefly present the most recent developments below.

Here we consider the important problem of *submodular function minimization*. This class of problems consists of unconstrained 0/1 programming problems

$$\min f(\mathbf{x}) : \mathbf{x} \in \{0, 1\}^n,$$

where the function f is submodular, i.e.,

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\max\{\mathbf{x}, \mathbf{y}\}) + f(\min\{\mathbf{x}, \mathbf{y}\}).$$

Here \max and \min denote the componentwise maximum and minimum of the vectors, respectively.

The fastest algorithm known for submodular function minimization seems to be by Orlin [54], who gave a strongly polynomial-time algorithm of running time $O(n^5 T_{\text{eval}} + n^6)$, where T_{eval} denotes the running time of the evaluation oracle. The algorithm is “combinatorial”, i.e., it does not use the ellipsoid method. This complexity bound simultaneously improved that of the fastest strongly polynomial-time algorithm using the ellipsoid method, of running time $\tilde{O}(n^5 T_{\text{eval}} + n^7)$ (see [50]) and the fastest “combinatorial” strongly polynomial-time algorithm by Iwata [35], of running time $O((n^6 T_{\text{eval}} + n^7) \log n)$. We remark that the fastest polynomial-time algorithm, by Iwata [35], runs in $O((n^4 T_{\text{eval}} + n^5) \log M)$, where M is the largest function value. We refer to the recent survey by Iwata [36], who reports on the developments that preceded Orlin's algorithm [54].

For the special case of *symmetric* submodular function minimization, i.e., $f(\mathbf{x}) = f(\mathbf{1} - \mathbf{x})$, Queyranne [56] presented an algorithm of running time $O(n^3 T_{\text{eval}})$.

Acknowledgments. The author wishes to thank the referees, in particular for their comments on the presentation of the Lenstra-type algorithm, and his student Robert Hildebrand for a subsequent discussion about this topic.

REFERENCES

- [1] L. ADLEMAN AND K. MANDERS, *Reducibility, randomness and intractability*, in Proc. 9th Annual ACM Symposium on Theory of Computing, 1977, pp. 151–163.
- [2] M. AGRAWAL, N. KAYAL, AND N. SAXENA, *PRIMES is in P*, Annals of Math., **160** (2004), pp. 781–793.
- [3] V. BALDONI, N. BERLINE, J.A. DE LOERA, M. KÖPPE, AND M. VERGNE, *How to integrate a polynomial over a simplex*: To appear in *Mathematics of Computation*. eprint arXiv:0809.2083 [math.MG], 2008.
- [4] B. BANK, J. HEINTZ, T. KRICK, R. MANDEL, AND P. SOLERNÓ, *Une borne optimale pour la programmation entière quasi-convexe*, Bull. Soc. math. France, **121** (1993), pp. 299–314.
- [5] B. BANK, T. KRICK, R. MANDEL, AND P. SOLERNÓ, *A geometrical bound for integer programming with polynomial constraints*, in Fundamentals of Computation Theory, Vol. **529** of Lecture Notes In Computer Science, Springer-Verlag, 1991, pp. 121–125.
- [6] A.I. BARVINOK, *A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*, Mathematics of Operations Research, **19** (1994), pp. 769–779.
- [7] A.I. BARVINOK AND J.E. POMMERSHEIM, *An algorithmic theory of lattice points in polyhedra*, in New Perspectives in Algebraic Combinatorics, L.J. Billera, A. Björner, C. Greene, R.E. Simion, and R.P. Stanley, eds., Vol. **38** of Math. Sci. Res. Inst. Publ., Cambridge Univ. Press, Cambridge, 1999, pp. 91–147.
- [8] S. BASU, R. POLLACK, AND M.-F. ROY, *Algorithms in Real Algebraic Geometry*, Springer-Verlag, second ed., 2006.
- [9] M. BELLARE AND P. ROGAWAY, *The complexity of approximating a nonlinear program*, in Pardalos [55].
- [10] M. BELLARE AND P. ROGAWAY, *The complexity of approximating a nonlinear program*, Mathematical Programming, **69** (1995), pp. 429–441.
- [11] Y. BERSTEIN, J. LEE, H. MARURI-AGUILAR, S. ONN, E. RICCOMAGNO, R. WEISMANTEL, AND H. WYNN, *Nonlinear matroid optimization and experimental design*, SIAM Journal on Discrete Mathematics, **22** (2008), pp. 901–919.
- [12] Y. BERSTEIN, J. LEE, S. ONN, AND R. WEISMANTEL, *Nonlinear optimization for matroid intersection and extensions*, IBM Research Report RC24610 (2008).
- [13] Y. BERSTEIN AND S. ONN, *Nonlinear bipartite matching*, Discrete Optimization, **5** (2008), pp. 53–65.
- [14] L. BLUM, M. SHUB, AND S. SMALE, *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines*, Bull. Am. Math. Soc., **21** (1989), pp. 1–46.
- [15] W.J. COOK, M.E. HARTMANN, R. KANNAN, AND C. McDIARMID, *On integer points in polyhedra*, Combinatorica, **12** (1992), pp. 27–37.
- [16] E. DE KLERK, *The complexity of optimizing over a simplex, hypercube or sphere: a short survey*, Central European Journal of Operations Research, **16** (2008), pp. 111–125.
- [17] E. DE KLERK, M. LAURENT, AND P.A. PARRILLO, *A PTAS for the minimization of polynomials of fixed degree over the simplex*, Theoretical Computer Science, **361** (2006), pp. 210–225.
- [18] J.A. DE LOERA, R. HEMMECKE, M. KÖPPE, AND R. WEISMANTEL, *FPTAS for mixed-integer polynomial optimization with a fixed number of variables*, in 17th ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 743–748.
- [19] ———, *Integer polynomial optimization in fixed dimension*, Mathematics of Operations Research, **31** (2006), pp. 147–153.
- [20] ———, *FPTAS for optimizing polynomials over the mixed-integer points of polytopes in fixed dimension*, Mathematical Programming, Series A, **118** (2008), pp. 273–290.

- [21] J.A. DE LOERA, R. HEMMECKE, S. ONN, AND R. WEISMANTEL, *N-fold integer programming*, Disc. Optim., to appear (2008).
- [22] J.A. DE LOERA AND S. ONN, *All linear and integer programs are slim 3-way transportation programs*, SIAM Journal of Optimization, **17** (2006), pp. 806–821.
- [23] ———, *Markov bases of three-way tables are arbitrarily complicated*, Journal of Symbolic Computation, **41** (2006), pp. 173–181.
- [24] F. EISENBRAND, *Integer programming and algorithmic geometry of numbers*, in 50 Years of Integer Programming 1958–2008, M. Jünger, T. Liebling, D. Naddef, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, eds., Springer-Verlag, 2010.
- [25] M.R. GAREY AND D.S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman and Company, New York, NY, 1979.
- [26] M. GRÖTSCHEL, L. LOVÁSZ, AND A. SCHRIJVER, *Geometric Algorithms and Combinatorial Optimization*, Springer, Berlin, Germany, 1988.
- [27] M.E. HARTMANN, *Cutting Planes and the Complexity of the Integer Hull*, phd thesis, Cornell University, Department of Operations Research and Industrial Engineering, Ithaca, NY, 1989.
- [28] J. HÅSTAD, *Some optimal inapproximability results*, in Proceedings of the 29th Symposium on the Theory of Computing (STOC), ACM, 1997, pp. 1–10.
- [29] J. HEINTZ, M. ROY, AND P. SOLERNÓ, *Sur la complexité du principe de Tarski-Seidenberg*, Bull. Soc. Math. France, **118** (1990), pp. 101–126.
- [30] S. HEINZ, *Complexity of integer quasiconvex polynomial optimization*, Journal of Complexity, **21** (2005), pp. 543–556.
- [31] R. HEMMECKE, M. KÖPPE, J. LEE, AND R. WEISMANTEL, *Nonlinear integer programming*, in 50 Years of Integer Programming 1958–2008, M. Jünger, T. Liebling, D. Naddef, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, eds., Springer-Verlag, 2010.
- [32] R. HEMMECKE, S. ONN, AND R. WEISMANTEL, *A polynomial oracle-time algorithm for convex integer minimization*, Mathematical Programming, Series A (2009). Published online 06 March 2009.
- [33] R. HILDEBRAND AND M. KÖPPE, *A faster algorithm for quasi-convex integer polynomial optimization*. eprint arXiv:1006.4661 [math.OC], 2010.
- [34] D. HOCHBAUM, *Complexity and algorithms for nonlinear optimization problems*, Annals of Operations Research, **153** (2007), pp. 257–296.
- [35] S. IWATA, *A faster scaling algorithm for minimizing submodular functions*, SIAM Journal on Computing, **32** (2003), pp. 833–840.
- [36] ———, *Submodular function minimization*, Mathematical Programming, **112** (2008), pp. 45–64.
- [37] R.G. JEROSLOW, *There cannot be any algorithm for integer programming with quadratic constraints*, Operations Research, **21** (1973), pp. 221–224.
- [38] J.P. JONES, *Universal diophantine equation*, Journal of Symbolic Logic, **47** (1982), pp. 403–410.
- [39] J.P. JONES AND YU. V. MATIYASEVICH, *Proof of recursive unsolvability of Hilbert’s tenth problem*, The American Mathematical Monthly, **98** (1991), pp. 689–709.
- [40] L.G. KHACHIAN, *Convexity and complexity in polynomial programming*, in Proceedings of the International Congress of Mathematicians, August 16–24, 1983, Warszawa, Z. Ciesielski and C. Olech, eds., New York, 1984, North-Holland, pp. 1569–1577.
- [41] L.G. KHACHIAN AND L. PORKOLAB, *Integer optimization on convex semialgebraic sets*, Discrete and Computational Geometry, **23** (2000), pp. 207–224.
- [42] J.C. LAGARIAS, *On the computational complexity of determining the solvability or unsolvability of the equation $x^2 - dy^2 = -1$* , Transactions of the American Mathematical Society, **260** (1980), pp. 485–508.

- [43] ———, *Succinct certificates for the solvability of binary quadratic diophantine equations*. e-print arXiv:math/0611209v1, 2006. Extended and updated version of a 1979 FOCS paper.
- [44] J. LEE, S. ONN, AND R. WEISMANTEL, *Nonlinear optimization over a weighted independence system*, IBM Research Report RC24513 (2008).
- [45] ———, *On test sets for nonlinear integer maximization*, Operations Research Letters, **36** (2008), pp. 439–443.
- [46] H.W. LENSTRA, *Integer programming with a fixed number of variables*, Mathematics of Operations Research, **8** (1983), pp. 538–548.
- [47] K. MANDERS AND L. ADLEMAN, *NP-complete decision problems for binary quadratics*, J. Comp. Sys. Sci., **16** (1978), pp. 168–184.
- [48] YU. V. MATIYASEVICH, *Enumerable sets are diophantine*, Doklady Akademii Nauk SSSR, **191** (1970), pp. 279–282. (Russian); English translation, Soviet Mathematics Doklady, Vol. **11** (1970), pp. 354–357.
- [49] ———, *Hilbert's tenth problem*, The MIT Press, Cambridge, MA, USA, 1993.
- [50] S.T. MCCORMICK, *Submodular function minimization*, in Discrete Optimization, K. Aardal, G. Nemhauser, and R. Weismantel, eds., Vol. **12** of Handbooks in Operations Research and Management Science, Elsevier, 2005.
- [51] T.S. MOTZKIN AND E.G. STRAUS, *Maxima for graphs and a new proof of a theorem of Turán*, Canadian Journal of Mathematics, **17** (1965), pp. 533–540.
- [52] W.T. OBUCHOWSKA, *On boundedness of (quasi-)convex integer optimization problems*, Math. Meth. Oper. Res., **68** (2008).
- [53] S. ONN, *Convex discrete optimization*. eprint arXiv:math/0703575, 2007.
- [54] J.B. ORLIN, *A faster strongly polynomial time algorithm for submodular function minimization*, Math. Program., Ser. A, **118** (2009), pp. 237–251.
- [55] P.M. PARDALOS, ed., *Complexity in Numerical Optimization*, World Scientific, 1993.
- [56] M. QUEYRANNE, *Minimizing symmetric submodular functions*, Mathematical Programming, **82** (1998), pp. 3–12.
- [57] J. RENEGAR, *On the computational complexity and geometry of the first-order theory of the reals, part I: Introduction. Preliminaries. The geometry of semi-algebraic sets. The decision problem for the existential theory of the reals*, Journal of Symbolic Computation, **13** (1992), pp. 255–300.
- [58] ———, *On the computational complexity and geometry of the first-order theory of the reals, part II: The general decision problem. Preliminaries for quantifier elimination*, Journal of Symbolic Computation, **13** (1992), pp. 301–328.
- [59] ———, *On the computational complexity and geometry of the first-order theory of the reals. part III: Quantifier elimination*, Journal of Symbolic Computation, **13** (1992), pp. 329–352.
- [60] ———, *On the computational complexity of approximating solutions for real algebraic formulae*, SIAM Journal on Computing, **21** (1992), pp. 1008–1025.
- [61] C.L. SIEGEL, *Zur Theorie der quadratischen Formen*, Nachrichten der Akademie der Wissenschaften in Göttingen, II, Mathematisch-Physikalische Klasse, **3** (1972), pp. 21–46.
- [62] T. SKOLEM, *Diophantische Gleichungen*, Vol. **5** of Ergebnisse der Mathematik und ihrer Grenzgebiete, 1938.
- [63] S.P. TARASOV AND L.G. KHACHIYAN, *Bounds of solutions and algorithmic complexity of systems of convex diophantine inequalities*, Soviet Math. Doklady, **22** (1980), pp. 700–704.
- [64] A.M. TURING, *On computable numbers, with an application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society, Series 2, **42** (1936), pp. 230–265. Errata in ibidem, **43** (1937):544–546.
- [65] S.A. VAVASIS, *Polynomial time weak approximation algorithms for quadratic programming*, in Pardalos [55].

THEORY AND APPLICATIONS OF N -FOLD INTEGER PROGRAMMING

SHMUEL ONN*

Abstract. We overview our recently introduced theory of n -fold integer programming which enables the polynomial time solution of fundamental linear and nonlinear integer programming problems in variable dimension. We demonstrate its power by obtaining the first polynomial time algorithms in several application areas including multicommodity flows and privacy in statistical databases.

Key words. Integer programming, transportation problem, multiway table, multicommodity flow, combinatorial optimization, nonlinear optimization, Graver base, Graver complexity, discrete optimization, privacy in databases, data security.

AMS(MOS) subject classifications. 05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C.

1. Introduction. Linear integer programming is the following fundamental optimization problem,

$$\min \{wx : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u\},$$

where A is an integer $m \times n$ matrix, $b \in \mathbb{Z}^m$, and $l, u \in \mathbb{Z}_\infty^n$ with $\mathbb{Z}_\infty := \mathbb{Z} \uplus \{\pm\infty\}$. It is generally NP-hard, but polynomial time solvable in two fundamental situations: the dimension is fixed [18]; the underlying matrix is totally unimodular [15].

Recently, in [4], a new fundamental polynomial time solvable situation was discovered. We proceed to describe this class of so-termed *n -fold integer programs*.

An $(r, s) \times t$ *bimatrix* is a matrix A consisting of two blocks A_1, A_2 , with A_1 its $r \times t$ submatrix consisting of the first r rows and A_2 its $s \times t$ submatrix consisting of the last s rows. The *n -fold product* of A is the following $(r + ns) \times nt$ matrix,

$$A^{(n)} := \begin{pmatrix} A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_2 \end{pmatrix}.$$

The following result of [4] asserts that n -fold integer programs are efficiently solvable.

*Technion - Israel Institute of Technology, 32000 Haifa, Israel (onn@ie.technion.ac.il). Supported in part by a grant from ISF - the Israel Science Foundation.

THEOREM 1.1. [4] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given positive integer n , bounds $l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and $w \in \mathbb{Z}^{nt}$, solves in time which is polynomial in n and in the binary-encoding length $\langle l, u, b, w \rangle$ of the rest of the data, the following so-called linear n -fold integer programming problem,*

$$\min \left\{ wx : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u \right\} .$$

Some explanatory notes are in order. First, the dimension of an n -fold integer program is nt and is variable. Second, n -fold products $A^{(n)}$ are highly non totally unimodular: the n -fold product of the simple $(0, 1) \times 1$ bimatrix with A_1 empty and $A_2 := 2$ satisfies $A^{(n)} = 2I_n$ and has exponential determinant 2^n . So this is indeed a class of programs which cannot be solved by methods of fixed dimension or totally unimodular matrices. Third, this class of programs turns out to be very natural and has numerous applications, the most generic being to integer optimization over multidimensional tables (see §2). In fact it is *universal*: the results of [7] imply that *every* integer program is an n -fold program over some simple bimatrix A (see §4).

The above theorem extends to n -fold integer programming with non-linear objective functions as well. The following results, from [12], [5] and [13], assert that the minimization and maximization of broad classes of convex functions over n -fold integer programs can also be done in polynomial time. The function f is presented either by a *comparison oracle* that for any two vectors x, y can answer whether or not $f(x) \leq f(y)$, or by an *evaluation oracle* that for any vector x can return $f(x)$.

In the next theorem, f is *separable convex*, namely $f(x) = \sum_i f_i(x_i)$ with each f_i univariate convex. Like linear forms, such functions can be minimized over totally unimodular programs [14]. We show that they can also be efficiently minimized over n -fold programs. The running time depends also on $\log \hat{f}$ with \hat{f} the maximum value of $|f(x)|$ over the feasible set (which need not be part of the input).

THEOREM 1.2. [12] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given $n, l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and separable convex $f : \mathbb{Z}^{nt} \rightarrow \mathbb{Z}$ presented by a comparison oracle, solves in time polynomial in n and $\langle l, u, b, \hat{f} \rangle$, the program*

$$\min \left\{ f(x) : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u \right\} .$$

An important natural special case of Theorem 1.2 is the following result that concerns finding a feasible point which is l_p -closest to a given desired goal point.

THEOREM 1.3. [12] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given positive integers n and p , $l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and $\hat{x} \in \mathbb{Z}^{nt}$, solves in time polynomial in n, p , and $\langle l, u, b, \hat{x} \rangle$, the following distance minimization program,*

$$\min \{ \|x - \hat{x}\|_p : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u \} . \quad (1.1)$$

For $p = \infty$ the problem (1.1) can be solved in time polynomial in n and $\langle l, u, b, \hat{x} \rangle$.

The next result concerns the *maximization* of a convex function of the composite form $f(Wx)$, with $f : \mathbb{Z}^d \rightarrow \mathbb{Z}$ convex and W an integer matrix with d rows.

THEOREM 1.4. [5] For each fixed d and $(r, s) \times t$ integer bimatrix A , there is an algorithm that, given n , bounds $l, u \in \mathbb{Z}_{\infty}^{nt}$, integer $d \times nt$ matrix W , $b \in \mathbb{Z}^{r+ns}$, and convex function $f : \mathbb{Z}^d \rightarrow \mathbb{R}$ presented by a comparison oracle, solves in time polynomial in n and $\langle W, l, u, b \rangle$, the convex n -fold integer maximization program

$$\max \{ f(Wx) : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u \} .$$

Finally, we have the following broad extension of Theorem 1.2 where the objective can include a composite term $f(Wx)$, with $f : \mathbb{Z}^d \rightarrow \mathbb{Z}$ separable convex and W an integer matrix with d rows, and where also inequalities on Wx can be included. As before, \hat{f}, \hat{g} denote the maximum values of $|f(Wx)|, |g(x)|$ over the feasible set.

THEOREM 1.5. [13] For each fixed integer $(r, s) \times t$ bimatrix A and integer $(p, q) \times t$ bimatrix W , there is an algorithm that, given $n, l, u \in \mathbb{Z}_{\infty}^{nt}$, $\hat{l}, \hat{u} \in \mathbb{Z}_{\infty}^{p+qn}$, $b \in \mathbb{Z}^{r+ns}$, and separable convex functions $f : \mathbb{Z}^{p+qn} \rightarrow \mathbb{Z}$, $g : \mathbb{Z}^{nt} \rightarrow \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial in n and $\langle l, u, \hat{l}, \hat{u}, b, \hat{f}, \hat{g} \rangle$, the generalized program

$$\min \left\{ f(W^{(n)}x) + g(x) : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, \hat{l} \leq W^{(n)}x \leq \hat{u}, l \leq x \leq u \right\} .$$

The article is organized as follows. In Section 2 we discuss some of the many applications of this theory and use Theorems 1.1–1.5 to obtain the first polynomial time algorithms for these applications. In Section 3 we provide a concise development of the theory of n -fold integer programming and prove our Theorems 1.1–1.5. Sections 2 and 3 can be read in any order. We conclude in Section 4 with a discussion of the universality of n -fold integer programming and of a new (di)-graph invariant, about which very little is known, that is important in understanding the complexity of our algorithms. Further discussion of n -fold integer programming within the broader context of *nonlinear discrete optimization* can be found in [21] and [22].

2. Applications.

2.1. Multiway tables. Multiway tables occur naturally in any context involving multiply-indexed variables. They have been studied extensively in mathematical programming in the context of high dimensional transportation problems (see [27, 28] and the references therein) and in

statistics in the context of disclosure control and privacy in statistical databases (see [3, 9] and the references therein). The theory of n -fold integer programming provides the first polynomial time algorithms for multiway table problems in these two contexts, which are discussed in §2.1.1 and §2.1.2 respectively.

We start with some terminology and background that will be used in the sequel. A *d-way table* is an $m_1 \times \cdots \times m_d$ array $x = (x_{i_1, \dots, i_d})$ of nonnegative integers. A *d-way transportation polytope* (*d-way polytope* for brevity) is the set of $m_1 \times \cdots \times m_d$ nonnegative arrays $x = (x_{i_1, \dots, i_d})$ with specified sums of entries over some of their lower dimensional subarrays (*margins* in statistics). The *d-way* tables with specified margins are the integer points in the *d-way* polytope. For example (see Figure 1), the 3-way polytope of $l \times m \times n$ arrays with specified line-sums (2-margins) is

$$T := \{x \in \mathbb{R}_+^{l \times m \times n} : \sum_i x_{i,j,k} = v_{*,j,k}, \sum_j x_{i,j,k} = v_{i,*,k}, \sum_k x_{i,j,k} = v_{i,j,*}\}$$

where the specified line-sums are $mn + ln + lm$ given nonnegative integer numbers

$$v_{*,j,k}, \quad v_{i,*,k}, \quad v_{i,j,*}, \quad 1 \leq i \leq l, \quad 1 \leq j \leq m, \quad 1 \leq k \leq n .$$

Our results hold for k -margins for any $0 \leq k \leq d$, and much more generally for any so-called *hierarchical family* of margins. For simplicity of the exposition, however, we restrict attention here to line-sums, that is, $(d - 1)$ -margins, only.

We conclude this preparation with the *universality theorem* for multiway tables and polytopes. It provides a powerful tool in establishing the presumable limits of polynomial time solvability of table problems, and will be used in §2.1.1 and §2.1.2 to contrast the polynomial time solvability attainable by n -fold integer programming.

THEOREM 2.1. [7] *Every rational polytope $P = \{y \in \mathbb{R}_+^d : Ay = b\}$ is in polynomial time computable integer preserving bijection with some $l \times m \times 3$ line-sum polytope*

$$T = \{x \in \mathbb{R}_+^{l \times m \times 3} : \sum_i x_{i,j,k} = v_{*,j,k}, \sum_j x_{i,j,k} = v_{i,*,k}, \sum_k x_{i,j,k} = v_{i,j,*}\}.$$

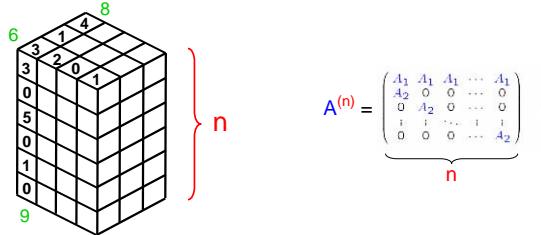
2.1.1. Multi-index transportation problems. The *multi-index transportation problem* of Motzkin [19] is the integer programming problem over multiway tables with specified margins. For line-sums it is the program

$$\begin{aligned} & \min\{wx : x \in \mathbb{Z}_+^{m_1 \times \cdots \times m_d}, \\ & \sum_{i_1} x_{i_1, \dots, i_d} = v_{*,i_2, \dots, i_d}, \dots, \sum_{i_d} x_{i_1, \dots, i_d} = v_{i_1, \dots, i_{d-1}, *}\}. \end{aligned}$$

For $d = 2$ this program is totally unimodular and can be solved in polynomial time. However, already for $d = 3$ it is generally not, and the problem

Multiway Tables

Consider $m_1 \times \dots \times m_d \times n$ tables with given margins such as line-sums:



Such tables form an n -fold program $\{x : A^{(n)}x = b, x \geq 0, x \text{ integer}\}$ for suitable bimatrix A determined by m_1, \dots, m_d where A_1 controls equations of margins which involve summation over layers, whereas A_2 controls equations of margins involving summation within a single layer at a time

FIG. 1. Multiway tables.

is much harder. Consider the problem over $l \times m \times n$ tables. If l, m, n are all fixed then the problem is solvable in polynomial time (in the natural binary-encoding length of the line-sums), but even in this very restricted situation one needs off-hand the algorithm of integer programming in fixed dimension lmn . If l, m, n are all variable then the problem is NP-hard [17]. The in-between cases are much more delicate and were resolved only recently. If two sides are variable and one is fixed then the problem is still NP-hard [6]; moreover, Theorem 2.1 implies that it is NP-hard even over $l \times m \times 3$ tables with fixed $n = 3$. Finally, if two sides are fixed and one is variable, then the problem can be solved in polynomial time by n -fold integer programming. Note that even over $3 \times 3 \times n$ tables, the only solution of the problem available to-date is the one given below using n -fold integer programming.

The polynomial time solvability of the multi-index transportation problem when one side is variable and the others are fixed extends to any dimension d . We have the following important result on the multi-index transportation problem.

THEOREM 2.2. [4] *For every fixed d, m_1, \dots, m_d , there is an algorithm that, given n , integer $m_1 \times \dots \times m_d \times n$ cost w , and integer line-sums $v =$*

$((v_{*,i_2,\dots,i_{d+1}}), \dots, (v_{i_1,\dots,i_d,*}))$, solves in time polynomial in n and $\langle w, v \rangle$, the $(d+1)$ -index transportation problem

$$\min\{wx : x \in \mathbb{Z}_+^{m_1 \times \dots \times m_d \times n},$$

$$\sum_{i_1} x_{i_1,\dots,i_{d+1}} = v_{*,i_2,\dots,i_{d+1}}, \dots, \sum_{i_{d+1}} x_{i_1,\dots,i_{d+1}} = v_{i_1,\dots,i_d,*}\}.$$

Proof. Re-index arrays as $x = (x^1, \dots, x^n)$ with each $x^{i_{d+1}} = (x_{i_1,\dots,i_d,i_{d+1}})$ a suitably indexed $m_1 m_2 \cdots m_d$ vector representing the i_{d+1} -th layer of x . Similarly re-index the array w . Let $t := r := m_1 m_2 \cdots m_d$ and $s := n(m_2 \cdots m_d + \cdots + m_1 \cdots m_{d-1})$. Let $b := (b^0, b^1, \dots, b^n) \in \mathbb{Z}^{r+ns}$, where $b^0 := (v_{i_1,\dots,i_d,*})$ and for $i_{d+1} = 1, \dots, n$,

$$b^{i_{d+1}} := ((v_{*,i_2,\dots,i_d,i_{d+1}}), \dots, (v_{i_1,\dots,i_{d-1},*,i_{d+1}})) .$$

Let A be the $(t, s) \times t$ bimatrix with first block $A_1 := I_t$ the $t \times t$ identity matrix and second block A_2 a matrix defining the line-sum equations on $m_1 \times \cdots \times m_d$ arrays. Then the equations $A_1(\sum_{i_{d+1}} x^{i_{d+1}}) = b^0$ represent the line-sum equations $\sum_{i_{d+1}} x_{i_1,\dots,i_{d+1}} = v_{i_1,\dots,i_d,*}$ where summations over layers occur, whereas the equations $A_2 x^{i_{d+1}} = b^{i_{d+1}}$ for $i_{d+1} = 1, \dots, n$ represent all other line-sum equations, where summations are within a single layer at a time. Therefore the multi-index transportation problem is encoded as the n -fold integer programming problem

$$\min \{wx : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, x \geq 0\} .$$

Using the algorithm of Theorem 1.1, this n -fold integer program, and hence the given multi-index transportation problem, can be solved in polynomial time. \square

This proof extends immediately to multi-index transportation problems with nonlinear objective functions of the forms in Theorems 1.2–1.5. Moreover, as mentioned before, a similar proof shows that multi-index transportation problems with k -margin constraints, and more generally, hierarchical margin constraints, can be encoded as n -fold integer programming problems as well. We state this as a corollary.

COROLLARY 2.1. [5] *For every fixed d and m_1, \dots, m_d , the nonlinear multi-index transportation problem, with any hierarchical margin constraints, over $(d+1)$ -way tables of format $m_1 \times \cdots \times m_d \times n$ with variable n layers, are polynomial time solvable.*

2.1.2. Privacy in statistical databases. A common practice in the disclosure of sensitive data contained in a multiway table is to release some of the table margins rather than the entries of the table. Once the margins are released, the security of any specific entry of the table is related to the set of possible values that can occur in that entry in all tables having the same margins as those of the source table in the database. In particular, if

this set consists of a unique value, that of the source table, then this entry can be exposed and privacy can be violated. This raises the following fundamental problem.

Entry uniqueness problem: Given hierarchical margin family and entry index, is the value which can occur in that entry in all tables with these margins, unique?

The complexity of this problem turns out to behave in analogy to the complexity of the multi-index transportation problem discussed in §2.1.1. Consider the problem for $d = 3$ over $l \times m \times n$ tables. It is polynomial time decidable when l, m, n are all fixed, and coNP-complete when l, m, n are all variable [17]. We discuss next in more detail the in-between cases which are more delicate and were settled only recently.

If two sides are variable and one is fixed then the problem is still coNP-complete, even over $l \times m \times 3$ tables with fixed $n = 3$ [20]. Moreover, Theorem 2.1 implies that *any set of nonnegative integers* is the set of values of an entry of some $l \times m \times 3$ tables with some specified line-sums. Figure 2 gives an example of line-sums for $6 \times 4 \times 3$ tables where one entry attains the set of values $\{0, 2\}$ which has a *gap*.

Set of Entry Values With a Gap

The **only values** occurring in the **designated entry** in all **$6 \times 4 \times 3$** tables with the **specified line-sums** are **0, 2**

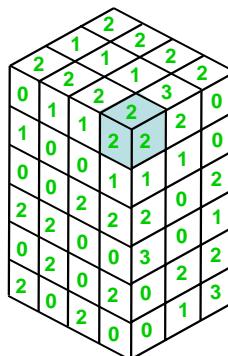


FIG. 2. Set of entry values with a gap.

THEOREM 2.3. [8] *For every finite set $S \subset \mathbb{Z}_+$ of nonnegative integers, there exist l, m , and line-sums for $l \times m \times 3$ tables, such that the set of values that occur in some fixed entry in all $l \times m \times 3$ tables that have these line-sums, is precisely S .*

Proof. Consider any finite set $S = \{s_1, \dots, s_h\} \subset \mathbb{Z}_+$. Consider the polytope

$$P := \{y \in \mathbb{R}_+^{h+1} : y_0 - \sum_{j=1}^h s_j y_j = 0, \sum_{j=1}^h y_j = 1\} .$$

By Theorem 2.1, there are l, m , and $l \times m \times 3$ polytope T with line-sums

$$v_{*,j,k}, \quad v_{i,*,k}, \quad v_{i,j,*}, \quad 1 \leq i \leq l, \quad 1 \leq j \leq m, \quad 1 \leq k \leq 3 ,$$

such that the integer points in T , which are precisely the $l \times m \times 3$ tables with these line-sums, are in bijection with the integer points in P . Moreover (see [7]), this bijection is obtained by a simple projection from $\mathbb{R}^{l \times m \times 3}$ to \mathbb{R}^{h+1} that erases all but some $h+1$ coordinates. Let $x_{i,j,k}$ be the coordinate that is mapped to y_0 . Then the set of values that this entry attains in all tables with these line-sums is, as desired,

$$\{x_{i,j,k} : x \in T \cap \mathbb{Z}^{l \times m \times 3}\} = \{y_0 : y \in P \cap \mathbb{Z}^{h+1}\} = S . \quad \square$$

Finally, if two sides are fixed and one is variable, then entry uniqueness can be decided in polynomial time by n -fold integer programming. Note that even over $3 \times 3 \times n$ tables, the only solution of the problem available to-date is the one below.

The polynomial time decidability of the problem when one side is variable and the others are fixed extends to any dimension d . It also extends to any hierarchical family of margins, but for simplicity we state it only for line-sums, as follows.

THEOREM 2.4. [20] *For every fixed d, m_1, \dots, m_d , there is an algorithm that, given n , integer line-sums $v = ((v_{*,i_2,\dots,i_{d+1}}), \dots, (v_{i_1,\dots,i_d,*}))$, and entry index (k_1, \dots, k_{d+1}) , solves in time which is polynomial in n and $\langle v \rangle$, the corresponding entry uniqueness problem, of deciding if the entry $x_{k_1,\dots,k_{d+1}}$ is the same in all $(d+1)$ -tables in the set*

$$S := \{x \in \mathbb{Z}_+^{m_1 \times \dots \times m_d \times n} : \sum_{i_1} x_{i_1,\dots,i_{d+1}} = v_{*,i_2,\dots,i_{d+1}}, \dots, \sum_{i_{d+1}} x_{i_1,\dots,i_{d+1}} = v_{i_1,\dots,i_d,*}\} .$$

Proof. By Theorem 2.2 we can solve in polynomial time both n -fold programs

$$l := \min \{x_{k_1,\dots,k_{d+1}} : x \in S\} ,$$

$$u := \max \{x_{k_1, \dots, k_{d+1}} : x \in S\} .$$

Clearly, entry $x_{k_1, \dots, k_{d+1}}$ has the same value in all tables with the given line-sums if and only if $l = u$, which can therefore be tested in polynomial time. \square

The algorithm of Theorem 2.4 and its extension to any family of hierarchical margins allow statistical agencies to efficiently check possible margins before disclosure: if an entry value is not unique then disclosure may be assumed secure, whereas if the value is unique then disclosure may be risky and fewer margins should be released.

We note that *long* tables, with one side much larger than the others, often arise in practical applications. For instance, in health statistical tables, the long factor may be the age of an individual, whereas other factors may be binary (yes-no) or ternary (subnormal, normal, and supnormal). Moreover, it is always possible to merge categories of factors, with the resulting coarser tables approximating the original ones, making the algorithm of Theorem 2.4 applicable.

Finally, we describe a procedure based on a suitable adaptation of the algorithm of Theorem 2.4, that constructs the entire set of values that can occur in a specified entry, rather than just decides its uniqueness. Here S is the set of tables satisfying the given (hierarchical) margins, and the running time is output-sensitive, that is, polynomial in the input encoding plus the number of elements in the output set.

Procedure for constructing the set of values in an entry:

1. Initialize $l := -\infty$, $u := \infty$, and $E := \emptyset$.
2. Solve in polynomial time the following linear n -fold integer programs:

$$\hat{l} := \min \{x_{k_1, \dots, k_{d+1}} : l \leq x_{k_1, \dots, k_{d+1}} \leq u, x \in S\} ,$$

$$\hat{u} := \max \{x_{k_1, \dots, k_{d+1}} : l \leq x_{k_1, \dots, k_{d+1}} \leq u, x \in S\} .$$

3. If the problems in Step 2 are feasible then update $l := \hat{l} + 1$, $u := \hat{u} - 1$, $E := E \cup \{\hat{l}, \hat{u}\}$, and repeat Step 2, else stop and output the set of values E .

2.2. Multicommodity flows. The multicommodity transshipment problem is a very general flow problem which seeks minimum cost routing of several discrete commodities over a digraph subject to vertex demand and edge capacity constraints. The data for the problem is as follows (see Figure 3 for a small example). There is a digraph G with s vertices and t edges. There are l types of commodities. Each commodity has a demand vector $d^k \in \mathbb{Z}^s$ with d_v^k the demand for commodity k at vertex v (interpreted as supply when positive and consumption when negative). Each edge e has a capacity u_e (upper bound on the combined flow of all commodities on

Multicommodity Transshipment Example

Data:

digraph G

two commodities: red and green

edge capacities u_e unlimited

edge costs $f_e(x^1_e + x^2_e) := (x^1_e + x^2_e)^2$ and $g^1_e(x^1_e) := g^2_e(x^2_e) := 0$

vertex demands:

$$d^1 := (3 \ -1 \ -2)$$

$$d^2 := (-3 \ 2 \ 1)$$

Solution:

$$X^1 = (3 \ 2 \ 0)$$

$$X^2 = (0 \ 2 \ 3)$$

Cost:

$$(3+0)^2 + (2+2)^2 + (0+3)^2 = 34$$

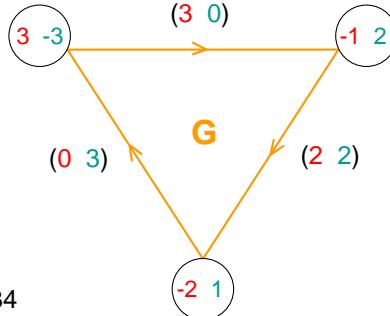


FIG. 3. Multicommodity transshipment example.

it). A *multicommodity transshipment* is a vector $x = (x^1, \dots, x^l)$ with $x^k \in \mathbb{Z}_+^t$ for all k and x_e^k the flow of commodity k on edge e , satisfying the capacity constraint $\sum_{k=1}^l x_e^k \leq u_e$ for each edge e and demand constraint $\sum_{e \in \delta^+(v)} x_e^k - \sum_{e \in \delta^-(v)} x_e^k = d_v^k$ for each vertex v and commodity k (with $\delta^+(v), \delta^-(v)$ the sets of edges entering and leaving vertex v).

The cost of transshipment x is defined as follows. There are cost functions $f_e, g_e^k : \mathbb{Z} \rightarrow \mathbb{Z}$ for each edge and each edge-commodity pair. The transshipment cost on edge e is $f_e(\sum_{k=1}^l x_e^k) + \sum_{k=1}^l g_e^k(x_e^k)$ with the first term being the value of f_e on the combined flow of all commodities on e and the second term being the sum of costs that depend on both the edge and the commodity. The total cost is

$$\sum_{e=1}^t \left(f_e \left(\sum_{k=1}^l x_e^k \right) + \sum_{k=1}^l g_e^k(x_e^k) \right) .$$

Our results apply to cost functions which can be standard linear or convex such as $\alpha_e |\sum_{k=1}^l x_e^k|^{\beta_e} + \sum_{k=1}^l \gamma_e^k |x_e^k|^{\delta_e^k}$ for some nonnegative integers $\alpha_e, \beta_e, \gamma_e^k, \delta_e^k$, which take into account the increase in cost due to channel congestion when subject to heavy traffic or communication load (with the linear case obtained by $\beta_e = \delta_e^k = 1$).

The theory of n -fold integer programming provides the first polynomial time algorithms for the problem in two broad situations discussed in §2.2.1 and §2.2.2.

2.2.1. The many-commodity transshipment problem. Here we consider the problem with *variable* number l of commodities over a fixed (but arbitrary) digraph - the so termed *many-commodity* transshipment problem. This problem may seem at first very restricted: however, even deciding if a feasible many-transshipment exists (regardless of its cost) is NP-complete already over the complete bipartite digraphs $K_{3,n}$ (oriented from one side to the other) with only 3 vertices on one side [13]; moreover, even over the single tiny digraph $K_{3,3}$, the only solution available to-date is the one given below via n -fold integer programming.

As usual, \hat{f} and \hat{g} denote the maximum absolute values of the objective functions f and g over the feasible set. It is usually easy to determine an upper bound on these values from the problem data. For instance, in the special case of linear cost functions f , g , bounds which are polynomial in the binary-encoding length of the costs α_e , γ_e^k , capacities u , and demands d_v^k , are easily obtained by Cramer's rule.

We have the following theorem on (nonlinear) many-commodity transshipment.

THEOREM 2.5. [13] *For every fixed digraph G there is an algorithm that, given l commodity types, demand $d_v^k \in \mathbb{Z}$ for each commodity k and vertex v , edge capacities $u_e \in \mathbb{Z}_+$, and convex costs $f_e, g_e^k : \mathbb{Z} \rightarrow \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial in l and $\langle d_v^k, u_e, \hat{f}, \hat{g} \rangle$, the many-commodity transshipment problem,*

$$\begin{aligned} & \min \sum_e \left(f_e \left(\sum_{k=1}^l x_e^k \right) + \sum_{k=1}^l g_e^k(x_e^k) \right) \\ \text{s.t. } & x_e^k \in \mathbb{Z}, \quad \sum_{e \in \delta^+(v)} x_e^k - \sum_{e \in \delta^-(v)} x_e^k = d_v^k, \quad \sum_{k=1}^l x_e^k \leq u_e, \quad x_e^k \geq 0 . \end{aligned}$$

Proof. Assume G has s vertices and t edges and let D be its $s \times t$ vertex-edge incidence matrix. Let $f : \mathbb{Z}^t \rightarrow \mathbb{Z}$ and $g : \mathbb{Z}^{lt} \rightarrow \mathbb{Z}$ be the separable convex functions defined by $f(y) := \sum_{e=1}^t f_e(y_e)$ with $y_e := \sum_{k=1}^l x_e^k$ and $g(x) := \sum_{e=1}^t \sum_{k=1}^l g_e^k(x_e^k)$. Let $x = (x^1, \dots, x^l)$ be the vector of variables with $x^k \in \mathbb{Z}^t$ the flow of commodity k for each k . Then the problem can be rewritten in vector form as

$$\min \left\{ f \left(\sum_{k=1}^l x^k \right) + g(x) : x \in \mathbb{Z}^{lt}, D x^k = d^k, \sum_{k=1}^l x^k \leq u, x \geq 0 \right\} .$$

We can now proceed in two ways.

First way: extend the vector of variables to $x = (x^0, x^1, \dots, x^l)$ with $x^0 \in \mathbb{Z}^t$ representing an additional slack commodity. Then the capacity

constraints become $\sum_{k=0}^l x^k = u$ and the cost function becomes $f(u - x_0) + g(x^1, \dots, x^l)$ which is also separable convex. Now let A be the $(t, s) \times t$ bimatrix with first block $A_1 := I_t$ the $t \times t$ identity matrix and second block $A_2 := D$. Let $d^0 := Du - \sum_{k=1}^l d^k$ and let $b := (u, d^0, d^1, \dots, d^l)$. Then the problem becomes the $(l+1)$ -fold integer program

$$\min \left\{ f(u - x^0) + g(x^1, \dots, x^l) : x \in \mathbb{Z}^{(l+1)t}, A^{(l)}x = b, x \geq 0 \right\}. \quad (2.1)$$

By Theorem 1.2 this program can be solved in polynomial time as claimed.

Second way: let A be the $(0, s) \times t$ bimatrix with first block A_1 empty and second block $A_2 := D$. Let W be the $(t, 0) \times t$ bimatrix with first block $W_1 := I_t$ the $t \times t$ identity matrix and second block W_2 empty. Let $b := (d^1, \dots, d^l)$. Then the problem is precisely the following l -fold integer program,

$$\min \left\{ f(W^{(l)}x) + g(x) : x \in \mathbb{Z}^{lt}, A^{(l)}x = b, W^{(l)}x \leq u, x \geq 0 \right\}.$$

By Theorem 1.5 this program can be solved in polynomial time as claimed. \square

We also point out the following immediate corollary of Theorem 2.5.

COROLLARY 2.2. *For any fixed s , the (convex) many-commodity transshipment problem with variable l commodities on any s -vertex digraph is polynomial time solvable.*

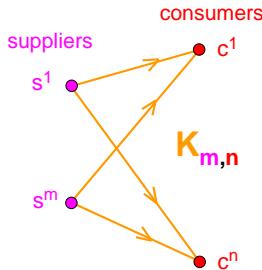
2.2.2. The multicommodity transportation problem. Here we consider the problem with fixed (but arbitrary) number l of commodities over any bipartite subdigraph of $K_{m,n}$ (oriented from one side to the other) - the so-called multicommodity *transportation* problem - with fixed number m of suppliers and *variable* number n of consumers. This is very natural in operations research applications where few facilities serve many customers. The problem is difficult even for $l = 2$ commodities: deciding if a feasible 2-commodity transportation exists (regardless of its cost) is NP-complete already over the complete bipartite digraphs $K_{m,n}$ [7]; moreover, even over the digraphs $K_{3,n}$ with only $m = 3$ suppliers, the only available solution to-date is the one given below via n -fold integer programming.

This problem seems harder than the one discussed in the previous subsection (with no seeming analog for non bipartite digraphs), and the formulation below is more delicate. Therefore it is convenient to change the labeling of the data a little bit as follows (see Figure 4). We now denote edges by pairs (i, j) where $1 \leq i \leq m$ is a supplier and $1 \leq j \leq n$ is a consumer. The demand vectors are now replaced by (nonnegative) supply and consumption vectors: each supplier i has a supply vector $s^i \in \mathbb{Z}_+^l$ with s_k^i its supply in commodity k , and each consumer j has a consumption vector $c^j \in \mathbb{Z}_+^l$ with c_k^j its consumption in commodity k . In addition, here each commodity k may have its own volume $v_k \in \mathbb{Z}_+$ per unit flow.

Multicommodity Transportation Problem

Find integer l commodity transportation x of minimum f, g cost from m suppliers to n consumers in the bipartite digraph $K_{m,n}$

Also given are supply and consumption vectors s^i and c^j in Z^l , edge capacities $u_{i,j}$ and volume v_k per unit commodity k



For suitable $(ml, l) \times ml$ bimatrix A and suitable $(0, m) \times ml$ bimatrix W derived from the v_k the problem becomes the n -fold integer program

$$\min \{ f(W^{(n)}x) + g(x) : x \in Z^{nl}, A^{(n)}x = (s^i, c^j), W^{(n)}x \leq u, x \geq 0 \}$$

FIG. 4. Multicommodity transportation problem.

A multicommodity transportation is now indexed as $x = (x^1, \dots, x^n)$ with $x^j = (x_{1,1}^j, \dots, x_{1,l}^j, \dots, x_{m,1}^j, \dots, x_{m,l}^j)$, where $x_{i,k}^j$ is the flow of commodity k from supplier i to consumer j . The capacity constraint on edge (i, j) is $\sum_{k=1}^l v_k x_{i,k}^j \leq u_{i,j}$ and the cost is $f_{i,j} \left(\sum_{k=1}^l v_k x_{i,k}^j \right) + \sum_{k=1}^l g_{i,k}^j (x_{i,k}^j)$ with $f_{i,j}, g_{i,k}^j : \mathbb{Z} \rightarrow \mathbb{Z}$ convex. As before, \hat{f}, \hat{g} denote the maximum absolute values of f, g over the feasible set.

We assume below that the underlying digraph is $K_{m,n}$ (with edges oriented from suppliers to consumers), since the problem over any subdigraph G of $K_{m,n}$ reduces to that over $K_{m,n}$ by simply forcing 0 capacity on all edges not present in G .

We have the following theorem on (nonlinear) multicommodity transportation.

THEOREM 2.6. [13] *For any fixed l commodities, m suppliers, and volumes v_k , there is an algorithm that, given n , supplies and demands $s^i, c^j \in \mathbb{Z}_+^l$, capacities $u_{i,j} \in \mathbb{Z}_+$, and convex costs $f_{i,j}, g_{i,k}^j : \mathbb{Z} \rightarrow \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial in n and $\langle s^i, c^j, u, \hat{f}, \hat{g} \rangle$, the multicommodity transportation problem,*

$$\begin{aligned} & \min \sum_{i,j} \left(f_{i,j} \left(\sum_k v_k x_{i,k}^j \right) + \sum_{k=1}^l g_{i,k}^j (x_{i,k}^j) \right) \\ \text{s.t. } & x_{i,k}^j \in \mathbb{Z}, \quad \sum_j x_{i,k}^j = s_k^i, \quad \sum_i x_{i,k}^j = c_k^j, \quad \sum_{k=1}^l v_k x_{i,k}^j \leq u_{i,j}, \quad x_{i,k}^j \geq 0 . \end{aligned}$$

Proof. Construct bimatrices A and W as follows. Let D be the $(l, 0) \times l$ bimatrix with first block $D_1 := I_l$ and second block D_2 empty. Let V be the $(0, 1) \times l$ bimatrix with first block V_1 empty and second block $V_2 := (v_1, \dots, v_l)$. Let A be the $(ml, l) \times ml$ bimatrix with first block $A_1 := I_{ml}$ and second block $A_2 := D^{(m)}$. Let W be the $(0, m) \times ml$ bimatrix with first block W_1 empty and second block $W_2 := V^{(m)}$. Let b be the $(ml + nl)$ -vector $b := (s^1, \dots, s^m, c^1, \dots, c^n)$.

Let $f : \mathbb{Z}^{nm} \rightarrow \mathbb{Z}$ and $g : \mathbb{Z}^{nml} \rightarrow \mathbb{Z}$ be the separable convex functions defined by $f(y) := \sum_{i,j} f_{i,j}(y_{i,j})$ with $y_{i,j} := \sum_{k=1}^l v_k x_{i,k}^j$ and $g(x) := \sum_{i,j} \sum_{k=1}^l g_{i,k}^j (x_{i,k}^j)$.

Now note that $A^{(n)}x$ is an $(ml + nl)$ -vector, whose first ml entries are the flows from each supplier of each commodity to all consumers, and whose last nl entries are the flows to each consumer of each commodity from all suppliers. Therefore the supply and consumption equations are encoded by $A^{(n)}x = b$. Next note that the nm -vector $y = (y_{1,1}, \dots, y_{m,1}, \dots, y_{1,n}, \dots, y_{m,n})$ satisfies $y = W^{(n)}x$. So the capacity constraints become $W^{(n)}x \leq u$ and the cost function becomes $f(W^{(n)}x) + g(x)$. Therefore, the problem is precisely the following n -fold integer program,

$$\min \left\{ f \left(W^{(n)}x \right) + g(x) : x \in \mathbb{Z}^{nml}, A^{(n)}x = b, W^{(n)}x \leq u, x \geq 0 \right\} .$$

By Theorem 1.5 this program can be solved in polynomial time as claimed. \square

3. Theory. In §3.1 we define Graver bases of integer matrices and show that they can be used to solve linear and nonlinear integer programs in polynomial time. In §3.2 we show that Graver bases of n -fold products can be computed in polynomial time and, incorporating the results of §3.1, prove our Theorems 1.1–1.5 that establish the polynomial time solvability of linear and nonlinear n -fold integer programming.

To simplify the presentation, and since the feasible set in most applications is finite or can be made finite by more careful modeling, whenever an algorithm detects that the feasible set is infinite, it simply stops. So, throughout our discussion, an algorithm is said to *solve a (nonlinear) integer programming problem* if it either finds an optimal solution x or concludes that the feasible set is infinite or empty.

As noted in the introduction, any nonlinear function f involved is presented either by a mere comparison oracle that for any two vectors x, y

can answer whether or not $f(x) \leq f(y)$, or by an evaluation oracle that for any vector x can return $f(x)$.

3.1. Graver bases and nonlinear integer programming. The *Graver basis* is a fundamental object in the theory of integer programming which was introduced by J. Graver already back in 1975 [11]. However, only very recently, in the series of papers [4, 5, 12], it was established that the Graver basis can be used to solve linear (as well as nonlinear) integer programming problems in polynomial time. In this subsection we describe these important new developments.

3.1.1. Graver bases. We begin with the definition of the Graver basis and some of its basic properties. Throughout this subsection let A be an integer $m \times n$ matrix. The *lattice* of A is the set $\mathcal{L}(A) := \{x \in \mathbb{Z}^n : Ax = 0\}$ of integer vectors in its kernel. We use $\mathcal{L}^*(A) := \{x \in \mathbb{Z}^n : Ax = 0, x \neq 0\}$ to denote the set of nonzero elements in $\mathcal{L}(A)$. We use a partial order \sqsubseteq on \mathbb{R}^n which extends the usual coordinate-wise partial order \leq on the nonnegative orthant \mathbb{R}_+^n and is defined as follows. For two vectors $x, y \in \mathbb{R}^n$ we write $x \sqsubseteq y$ and say that x is *conformal* to y if $x_i y_i \geq 0$ and $|x_i| \leq |y_i|$ for $i = 1, \dots, n$, that is, x and y lie in the same orthant of \mathbb{R}^n and each component of x is bounded by the corresponding component of y in absolute value. A suitable extension of the classical lemma of Gordan [10] implies that every subset of \mathbb{Z}^n has finitely many \sqsubseteq -minimal elements. We have the following fundamental definition.

DEFINITION 3.1. [11] The *Graver basis* of an integer matrix A is defined to be the finite set $\mathcal{G}(A) \subset \mathbb{Z}^n$ of \sqsubseteq -minimal elements in $\mathcal{L}^*(A) = \{x \in \mathbb{Z}^n : Ax = 0, x \neq 0\}$. Note that $\mathcal{G}(A)$ is centrally symmetric, that is, $g \in \mathcal{G}(A)$ if and only if $-g \in \mathcal{G}(A)$. For instance, the Graver basis of the 1×3 matrix $A := [1 \ 2 \ 1]$ consists of 8 elements,

$$\mathcal{G}(A) = \pm \{(2, -1, 0), (0, -1, 2), (1, 0, -1), (1, -1, 1)\} .$$

Note also that the Graver basis may contain elements, such as $(1, -1, 1)$ in the above small example, whose support involves linearly dependent columns of A . So the cardinality of the Graver basis cannot be bounded in terms of m and n only and depends on the entries of A as well. Indeed, the Graver basis is typically exponential and cannot be written down, let alone computed, in polynomial time. But, as we will show in the next section, for n -fold products it can be computed efficiently.

A finite sum $u := \sum_i v_i$ of vectors in \mathbb{R}^n is called *conformal* if $v_i \sqsubseteq u$ for all i and hence all summands lie in the same orthant. We start with a simple lemma.

LEMMA 3.1. *Any $x \in \mathcal{L}^*(A)$ is a conformal sum $x = \sum_i g_i$ of Graver basis elements $g_i \in \mathcal{G}(A)$, with some elements possibly appearing more than once in the sum.*

Proof. We use induction on the well partial order \sqsubseteq . Consider any $x \in \mathcal{L}^*(A)$. If it is \sqsubseteq -minimal in $\mathcal{L}^*(A)$ then $x \in \mathcal{G}(A)$ by definition of the

Graver basis and we are done. Otherwise, there is an element $g \in \mathcal{G}(A)$ such that $g \sqsubset x$. Set $y := x - g$. Then $y \in \mathcal{L}^*(A)$ and $y \sqsubset x$, so by induction there is a conformal sum $y = \sum_i g_i$ with $g_i \in \mathcal{G}(A)$ for all i . Now $x = g + \sum_i g_i$ is a conformal sum of x . \square

We now provide a stronger form of Lemma 3.1 which basically follows from the integer analogs of Carathéodory's theorem established in [2] and [26].

LEMMA 3.2. *Any $x \in \mathcal{L}^*(A)$ is a conformal sum $x = \sum_{i=1}^t \lambda_i g_i$ involving $t \leq 2n - 2$ Graver basis elements $g_i \in \mathcal{G}(A)$ with nonnegative integer coefficients $\lambda_i \in \mathbb{Z}_+$.*

Proof. We prove the slightly weaker bound $t \leq 2n - 1$ from [2]. A proof of the stronger bound can be found in [26]. Consider any $x \in \mathcal{L}^*(A)$ and let g_1, \dots, g_s be all elements of $\mathcal{G}(A)$ lying in the same orthant as x . Consider the linear program

$$\max \left\{ \sum_{i=1}^s \lambda_i : x = \sum_{i=1}^s \lambda_i g_i, \lambda_i \in \mathbb{R}_+ \right\}. \quad (3.1)$$

By Lemma 3.1 the point x is a nonnegative linear combination of the g_i and hence the program (3.1) is feasible. Since all g_i are nonzero and in the same orthant as x , program (3.1) is also bounded. As is well known, it then has a *basic* optimal solution, that is, an optimal solution $\lambda_1, \dots, \lambda_s$ with at most n of the λ_i nonzero. Let

$$y := \sum (\lambda_i - \lfloor \lambda_i \rfloor) g_i = x - \sum \lfloor \lambda_i \rfloor g_i.$$

If $y = 0$ then $x = \sum \lfloor \lambda_i \rfloor g_i$ is a conformal sum of at most n of the g_i and we are done. Otherwise, $y \in \mathcal{L}^*(A)$ and y lies in the same orthant as x , and hence, by Lemma 3.1 again, $y = \sum_{i=1}^s \mu_i g_i$ with all $\mu_i \in \mathbb{Z}_+$. Then $x = \sum (\mu_i + \lfloor \lambda_i \rfloor) g_i$ and hence, since the λ_i form an optimal solution to (3.1), we have $\sum (\mu_i + \lfloor \lambda_i \rfloor) \leq \sum \lambda_i$. Therefore $\sum \mu_i \leq \sum (\lambda_i - \lfloor \lambda_i \rfloor) < n$ with the last inequality holding since at most n of the λ_i are nonzero. Since the μ_i are integer, at most $n-1$ of them are nonzero. So $x = \sum (\mu_i + \lfloor \lambda_i \rfloor) g_i$ is a conformal sum of x involving at most $2n - 1$ of the g_i . \square

The Graver basis also enables to check the finiteness of a feasible integer program.

LEMMA 3.3. *Let $\mathcal{G}(A)$ be the Graver basis of matrix A and let $l, u \in \mathbb{Z}_{\infty}^n$. If there is some $g \in \mathcal{G}(A)$ satisfying $g_i \leq 0$ whenever $u_i < \infty$ and $g_i \geq 0$ whenever $l_i > -\infty$ then every set of the form $S := \{x \in \mathbb{Z}^n : Ax = b, l \leq x \leq u\}$ is either empty or infinite, whereas if there is no such g , then every set S of this form is finite. Clearly, the existence of such g can be checked in time polynomial in $\langle \mathcal{G}(A), l, u \rangle$.*

Proof. First suppose there exists such g . Consider any such S . Suppose S contains some point x . Then for all $\lambda \in \mathbb{Z}_+$ we have $l \leq x + \lambda g \leq u$ and $A(x + \lambda g) = Ax = b$ and hence $x + \lambda g \in S$, so S is infinite. Next suppose

S is infinite. Then the polyhedron $P := \{x \in \mathbb{R}^n : Ax = b, l \leq x \leq u\}$ is unbounded and hence, as is well known, has a recession vector, that is, a nonzero h , which we may assume to be integer, such that $x + \alpha h \in P$ for all $x \in P$ and $\alpha \geq 0$. This implies that $h \in \mathcal{L}^*(A)$ and that $h_i \leq 0$ whenever $u_i < \infty$ and $h_i \geq 0$ whenever $l_i > -\infty$. So h is a conformal sum $h = \sum g_i$ of vectors $g_i \in \mathcal{G}(A)$, each of which also satisfies $g_i \leq 0$ whenever $u_i < \infty$ and $g_i \geq 0$ whenever $l_i > -\infty$, providing such g . \square

3.1.2. Separable convex integer minimization. In this subsection we consider the following nonlinear integer minimization problem

$$\min\{f(x) : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u\}, \quad (3.2)$$

where A is an integer $m \times n$ matrix, $b \in \mathbb{Z}^m$, $l, u \in \mathbb{Z}_\infty^n$, and $f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ is a separable convex function, that is, $f(x) = \sum_{j=1}^n f_j(x_j)$ with $f_j : \mathbb{Z} \rightarrow \mathbb{Z}$ a univariate convex function for all j . We prove a sequence of lemmas and then combine them to show that the Graver basis of A enables to solve this problem in polynomial time.

We start with two simple lemmas about univariate convex functions. The first lemma establishes a certain *supermodularity* property of such functions.

LEMMA 3.4. *Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be a univariate convex function, let r be a real number, and let s_1, \dots, s_m be real numbers satisfying $s_i s_j \geq 0$ for all i, j . Then we have*

$$f\left(r + \sum_{i=1}^m s_i\right) - f(r) \geq \sum_{i=1}^m (f(r + s_i) - f(r)).$$

Proof. We use induction on m . The claim holding trivially for $m = 1$, consider $m > 1$. Since all nonzero s_i have the same sign, $s_m = \lambda \sum_{i=1}^m s_i$ for some $0 \leq \lambda \leq 1$. Then

$$r + s_m = (1-\lambda)r + \lambda \left(r + \sum_{i=1}^{m-1} s_i\right), \quad r + \sum_{i=1}^{m-1} s_i = \lambda r + (1-\lambda) \left(r + \sum_{i=1}^m s_i\right),$$

and so the convexity of f implies

$$\begin{aligned} & f(r + s_m) + f\left(r + \sum_{i=1}^{m-1} s_i\right) \\ & \leq (1-\lambda)f(r) + \lambda f\left(r + \sum_{i=1}^{m-1} s_i\right) + \lambda f(r) + (1-\lambda)f\left(r + \sum_{i=1}^m s_i\right) \\ & = f(r) + f\left(r + \sum_{i=1}^m s_i\right). \end{aligned}$$

Subtracting $2f(r)$ from both sides and applying induction, we obtain, as claimed,

$$\begin{aligned} f\left(r + \sum_{i=1}^m s_i\right) - f(r) \\ \geq f(r + s_m) - f(r) + f\left(r + \sum_{i=1}^{m-1} s_i\right) - f(r) \\ \geq \sum_{i=1}^m (f(r + s_i) - f(r)) . \quad \square \end{aligned}$$

The second lemma shows that univariate convex functions can be minimized efficiently over an interval of integers using repeated bisections.

LEMMA 3.5. *There is an algorithm that, given any two integer numbers $r \leq s$ and any univariate convex function $f : \mathbb{Z} \rightarrow \mathbb{R}$ given by a comparison oracle, solves in time polynomial in $\langle r, s \rangle$ the following univariate integer minimization problem,*

$$\min \{ f(\lambda) : \lambda \in \mathbb{Z}, r \leq \lambda \leq s \} .$$

Proof. If $r = s$ then $\lambda := r$ is optimal. Assume then $r \leq s - 1$. Consider the integers

$$r \leq \left\lfloor \frac{r+s}{2} \right\rfloor < \left\lfloor \frac{r+s}{2} \right\rfloor + 1 \leq s .$$

Use the oracle of f to compare $f\left(\left\lfloor \frac{r+s}{2} \right\rfloor\right)$ and $f\left(\left\lfloor \frac{r+s}{2} \right\rfloor + 1\right)$. By the convexity of f :

$$\begin{aligned} f\left(\left\lfloor \frac{r+s}{2} \right\rfloor\right) = f\left(\left\lfloor \frac{r+s}{2} \right\rfloor + 1\right) &\Rightarrow \lambda := \left\lfloor \frac{r+s}{2} \right\rfloor \text{ is a minimum of } f; \\ f\left(\left\lfloor \frac{r+s}{2} \right\rfloor\right) < f\left(\left\lfloor \frac{r+s}{2} \right\rfloor + 1\right) &\Rightarrow \text{the minimum of } f \text{ is in } [r, \left\lfloor \frac{r+s}{2} \right\rfloor]; \\ f\left(\left\lfloor \frac{r+s}{2} \right\rfloor\right) > f\left(\left\lfloor \frac{r+s}{2} \right\rfloor + 1\right) &\Rightarrow \text{the minimum of } f \text{ is in } [\left\lfloor \frac{r+s}{2} \right\rfloor + 1, s]. \end{aligned}$$

Thus, we either obtain the optimal point, or bisect the interval $[r, s]$ and repeat. So in $O(\log(s-r)) = O(\langle r, s \rangle)$ bisections we find an optimal solution $\lambda \in \mathbb{Z} \cap [r, s]$. \square

The next two lemmas extend Lemmas 3.4 and 3.5. The first lemma shows the supermodularity of separable convex functions with respect to conformal sums.

LEMMA 3.6. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be any separable convex function, let $x \in \mathbb{R}^n$ be any point, and let $\sum g_i$ be any conformal sum in \mathbb{R}^n . Then the following inequality holds,*

$$f\left(x + \sum g_i\right) - f(x) \geq \sum (f(x + g_i) - f(x)) .$$

Proof. Let $f_j : \mathbb{R} \rightarrow \mathbb{R}$ be univariate convex functions such that $f(x) = \sum_{j=1}^n f_j(x_j)$. Consider any $1 \leq j \leq n$. Since $\sum g_i$ is a conformal

sum, we have $g_{i,j}g_{k,j} \geq 0$ for all i, k and so, setting $r := x_j$ and $s_i := g_{i,j}$ for all i , Lemma 3.4 applied to f_j implies

$$f_j \left(x_j + \sum_i g_{i,j} \right) - f_j(x_j) \geq \sum_i (f_j(x_j + g_{i,j}) - f_j(x_j)) . \quad (3.3)$$

Summing the equations (3.3) for $j = 1, \dots, n$, we obtain the claimed inequality. \square

The second lemma concerns finding a best improvement step in a given direction.

LEMMA 3.7. *There is an algorithm that, given bounds $l, u \in \mathbb{Z}_\infty^n$, direction $g \in \mathbb{Z}^n$, point $x \in \mathbb{Z}^n$ with $l \leq x \leq u$, and convex function $f : \mathbb{Z}^n \rightarrow \mathbb{R}$ presented by comparison oracle, solves in time polynomial in $\langle l, u, g, x \rangle$, the univariate problem,*

$$\min\{f(x + \lambda g) : \lambda \in \mathbb{Z}_+, l \leq x + \lambda g \leq u\} \quad (3.4)$$

Proof. Let $S := \{\lambda \in \mathbb{Z}_+ : l \leq x + \lambda g \leq u\}$ be the feasible set and let $s := \sup S$, which is easy to determine. If $s = \infty$ then conclude that S is infinite and stop. Otherwise, $S = \{0, 1, \dots, s\}$ and the problem can be solved by the algorithm of Lemma 3.5 minimizing the univariate convex function $h(\lambda) := h(x + \lambda g)$ over S . \square

We can now show that the Graver basis of A allows to solve problem (3.2) in polynomial time, provided we are given an initial feasible point to start with. We will later show how to find such an initial point as well. As noted in the introduction, \hat{f} below denotes the maximum value of $|f(x)|$ over the feasible set (which need not be part of the input). An outline of the algorithm is provided in [Figure 5](#).

LEMMA 3.8. *There is an algorithm that, given an integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, vectors $l, u \in \mathbb{Z}_\infty^n$ and $x \in \mathbb{Z}^n$ with $l \leq x \leq u$, and separable convex function $f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ presented by a comparison oracle, solves the integer program*

$$\min\{f(z) : z \in \mathbb{Z}^n, Az = b, l \leq z \leq u\} , \quad b := Ax , \quad (3.5)$$

in time polynomial in the binary-encoding length $\langle \mathcal{G}(A), l, u, x, \hat{f} \rangle$ of the data.

Proof. First, apply the algorithm of Lemma 3.3 to $\mathcal{G}(A)$ and l, u and either detect that the feasible set is infinite and stop, or conclude it is finite and continue. Next produce a sequence of feasible points x_0, x_1, \dots, x_s with $x_0 := x$ the given input point, as follows. Having obtained x_k , solve the univariate minimization problem

$$\min\{f(x_k + \lambda g) : \lambda \in \mathbb{Z}_+, g \in \mathcal{G}(A), l \leq x_k + \lambda g \leq u\} \quad (3.6)$$

by applying the algorithm of Lemma 3.7 for each $g \in \mathcal{G}(A)$. If the minimal value in (3.6) satisfies $f(x_k + \lambda g) < f(x_k)$ then set $x_{k+1} := x_k + \lambda g$

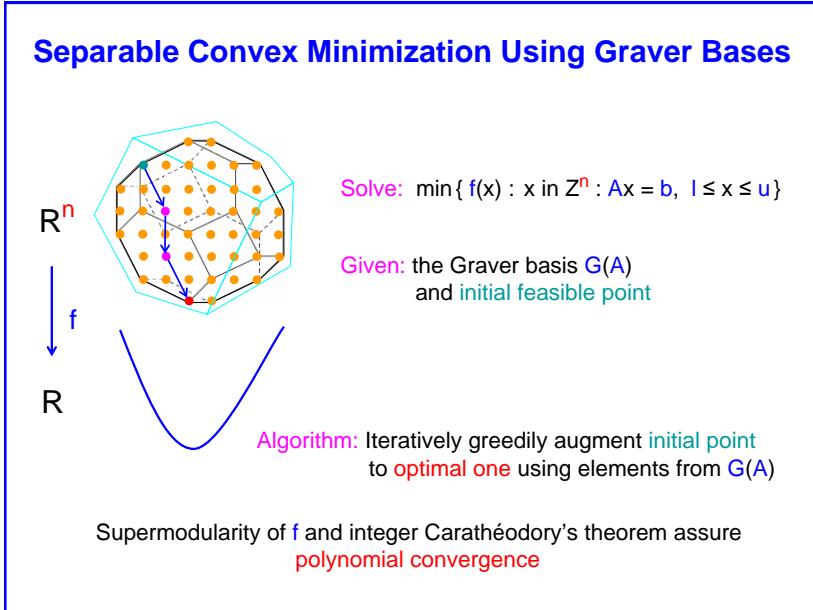


FIG. 5. Separable convex minimization using Graver bases.

and repeat, else stop and output the last point x_s in the sequence. Now, $Ax_{k+1} = A(x_k + \lambda g) = Ax_k = b$ by induction on k , so each x_k is feasible. Since the feasible set is finite and the x_k have decreasing objective values and hence distinct, the algorithm terminates.

We now show that the point x_s output by the algorithm is optimal. Let x^* be any optimal solution to (3.5). Consider any point x_k in the sequence and suppose it is not optimal. We claim that a new point x_{k+1} will be produced and will satisfy

$$f(x_{k+1}) - f(x^*) \leq \frac{2n-3}{2n-2} (f(x_k) - f(x^*)). \quad (3.7)$$

By Lemma 3.2, we can write the difference $x^* - x_k = \sum_{i=1}^t \lambda_i g_i$ as conformal sum involving $1 \leq t \leq 2n-2$ elements $g_i \in \mathcal{G}(A)$ with all $\lambda_i \in \mathbb{Z}_+$. By Lemma 3.6,

$$f(x^*) - f(x_k) = f\left(x_k + \sum_{i=1}^t \lambda_i g_i\right) - f(x_k) \geq \sum_{i=1}^t (f(x_k + \lambda_i g_i) - f(x_k)).$$

Adding $t(f(x_k) - f(x^*))$ on both sides and rearranging terms we obtain

$$\sum_{i=1}^t (f(x_k + \lambda_i g_i) - f(x^*)) \leq (t-1)(f(x_k) - f(x^*)) .$$

Therefore there is some summand on the left-hand side satisfying

$$f(x_k + \lambda_i g_i) - f(x^*) \leq \frac{t-1}{t} (f(x_k) - f(x^*)) \leq \frac{2n-3}{2n-2} (f(x_k) - f(x^*)) .$$

So the point $x_k + \lambda g$ attaining minimum in (3.6) satisfies

$$f(x_k + \lambda g) - f(x^*) \leq f(x_k + \lambda_i g_i) - f(x^*) \leq \frac{2n-3}{2n-2} (f(x_k) - f(x^*))$$

and so indeed $x_{k+1} := x_k + \lambda g$ will be produced and will satisfy (3.7). This shows that the last point x_s produced and output by the algorithm is indeed optimal.

We proceed to bound the number s of points. Consider any $i < s$ and the intermediate non optimal point x_i in the sequence produced by the algorithm. Then $f(x_i) > f(x^*)$ with both values integer, and so repeated use of (3.7) gives

$$\begin{aligned} 1 \leq f(x_i) - f(x^*) &= \prod_{k=0}^{i-1} \frac{f(x_{k+1}) - f(x^*)}{f(x_k) - f(x^*)} (f(x) - f(x^*)) \\ &\leq \left(\frac{2n-3}{2n-2} \right)^i (f(x) - f(x^*)) \end{aligned}$$

and therefore

$$i \leq \left(\log \frac{2n-2}{2n-3} \right)^{-1} \log (f(x) - f(x^*)) .$$

Therefore the number s of points produced by the algorithm is at most one unit larger than this bound, and using a simple bound on the logarithm, we obtain

$$s = O(n \log(f(x) - f(x^*))) .$$

Thus, the number of points produced and the total running time are polynomial. \square

Next we show that Lemma 3.8 can also be used to find an initial feasible point for the given integer program or assert that none exists in polynomial time.

LEMMA 3.9. *There is an algorithm that, given integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}_\infty^n$, and $b \in \mathbb{Z}^m$, either finds an $x \in \mathbb{Z}^n$*

satisfying $l \leq x \leq u$ and $Ax = b$ or asserts that none exists, in time which is polynomial in $\langle A, \mathcal{G}(A), l, u, b \rangle$.

Proof. Assume that $l \leq u$ and that $l_i < \infty$ and $u_j > -\infty$ for all j , since otherwise there is no feasible point. Also assume that there is no $g \in \mathcal{G}(A)$ satisfying $g_i \leq 0$ whenever $u_i < \infty$ and $g_i \geq 0$ whenever $l_i > -\infty$, since otherwise S is empty or infinite by Lemma 3.3. Now, either detect there is no integer solution to the system of equations $Ax = b$ (without the lower and upper bound constraints) and stop, or determine some such solution $\hat{x} \in \mathbb{Z}^n$ and continue; it is well known that this can be done in polynomial time, say, using the Hermite normal form of A , see [25]. Next define a separable convex function on \mathbb{Z}^n by $f(x) := \sum_{j=1}^n f_j(x_j)$ with

$$f_j(x_j) := \begin{cases} l_j - x_j, & \text{if } x_j < l_j \\ 0, & \text{if } l_j \leq x_j \leq u_j \\ x_j - u_j, & \text{if } x_j > u_j \end{cases}, \quad j = 1, \dots, n$$

and extended lower and upper bounds

$$\hat{l}_j := \min\{l_j, \hat{x}_j\}, \quad \hat{u}_j := \max\{u_j, \hat{x}_j\}, \quad j = 1, \dots, n.$$

Consider the auxiliary separable convex integer program

$$\min\{f(z) : z \in \mathbb{Z}^n, Az = b, \hat{l} \leq z \leq \hat{u}\}. \quad (3.8)$$

First note that $\hat{l}_j > -\infty$ if and only if $l_j > -\infty$ and $\hat{u}_j < \infty$ if and only if $u_j < \infty$. Therefore there is no $g \in \mathcal{G}(A)$ satisfying $g_i \leq 0$ whenever $\hat{u}_i < \infty$ and $g_i \geq 0$ whenever $\hat{l}_i > -\infty$ and hence the feasible set of (3.8) is finite by Lemma 3.3. Next note that \hat{x} is feasible in (3.8). Now apply the algorithm of Lemma 3.8 to (3.8) and obtain an optimal solution x . Note that this can be done in polynomial time since the binary length of \hat{x} and therefore also of \hat{l} , \hat{u} and of the maximum value \hat{f} of $|f(x)|$ over the feasible set of (3.8) are polynomial in the length of the data.

Now note that every point $z \in S$ is feasible in (3.8), and every point z feasible in (3.8) satisfies $f(z) \geq 0$ with equality if and only if $z \in S$. So, if $f(x) > 0$ then the original set S is empty, whereas if $f(x) = 0$ then $x \in S$ is a feasible point. \square

We are finally in position, using Lemmas 3.8 and 3.9, to show that the Graver basis allows to solve the nonlinear integer program (3.2) in polynomial time. As usual, \hat{f} is the maximum of $|f(x)|$ over the feasible set and need not be part of the input.

THEOREM 3.1. [12] *There is an algorithm that, given integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}_{\infty}^n$, $b \in \mathbb{Z}^m$, and separable convex $f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ presented by comparison oracle, solves in time polynomial in $\langle A, \mathcal{G}(A), l, u, b, \hat{f} \rangle$ the problem*

$$\min\{f(x) : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u\}.$$

Proof. First, apply the polynomial time algorithm of Lemma 3.9 and either conclude that the feasible set is infinite or empty and stop, or obtain an initial feasible point and continue. Next, apply the polynomial time algorithm of Lemma 3.8 and either conclude that the feasible set is infinite or obtain an optimal solution. \square

3.1.3. Specializations and extensions.

LINEAR INTEGER PROGRAMMING. Clearly, any linear function $wx = \sum_{i=1}^n w_i x_i$ is separable convex. Moreover, an upper bound on $|wx|$ over the feasible set (when finite), which is polynomial in the binary-encoding length of the data, readily follows from Cramer's rule. Therefore we obtain, as an immediate special case of Theorem 3.1, the following important result, asserting that Graver bases enable the polynomial time solution of linear integer programming.

THEOREM 3.2. [4] *There is an algorithm that, given an integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}_{\infty}^n$, $b \in \mathbb{Z}^m$, and $w \in \mathbb{Z}^n$, solves in time which is polynomial in $\langle A, \mathcal{G}(A), l, u, b, w \rangle$, the following linear integer programming problem,*

$$\min\{wx : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u\}.$$

DISTANCE MINIMIZATION. Another useful special case of Theorem 3.1 which is natural in various applications such as image processing, tomography, communication, and error correcting codes, is the following result, which asserts that the Graver basis enables to determine a feasible point which is l_p -closest to a given desired goal point in polynomial time.

THEOREM 3.3. [12] *There is an algorithm that, given integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, positive integer p , vectors $l, u \in \mathbb{Z}_{\infty}^n$, $b \in \mathbb{Z}^m$, and $\hat{x} \in \mathbb{Z}^n$, solves in time polynomial in p and $\langle A, \mathcal{G}(A), l, u, b, \hat{x} \rangle$, the distance minimization problem*

$$\min \{\|x - \hat{x}\|_p : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u\}. \quad (3.9)$$

For $p = \infty$ the problem (3.9) can be solved in time which is polynomial in $\langle A, \mathcal{G}(A), l, u, b, \hat{x} \rangle$.

Proof. For finite p apply the algorithm of Theorem 3.1 taking f to be the p -th power $\|x - \hat{x}\|_p^p$ of the l_p distance. If the feasible set is nonempty and finite (else the algorithm stops) then the maximum value \hat{f} of $|f(x)|$ over it is polynomial in p and $\langle A, l, u, b, \hat{x} \rangle$, and hence an optimal solution can be found in polynomial time.

Consider $p = \infty$. Using Cramer's rule it is easy to compute an integer ρ with $\langle \rho \rangle$ polynomially bounded in $\langle A, l, u, b \rangle$ that, if the feasible set is finite, provides an upper bound on $\|x\|_{\infty}$ for any feasible x . Let q be a positive integer satisfying

$$q > \frac{\log n}{\log(1 + (2\rho)^{-1})}.$$

Now apply the algorithm of the first paragraph above for the l_q distance. Assuming the feasible set is nonempty and finite (else the algorithm stops) let x^* be the feasible point which minimizes the l_q distance to \hat{x} obtained by the algorithm. We claim that it also minimizes the l_∞ distance to \hat{x} and hence is the desired optimal solution. Consider any feasible point x . By standard inequalities between the l_∞ and l_q norms,

$$\|x^* - \hat{x}\|_\infty \leq \|x^* - \hat{x}\|_q \leq \|x - \hat{x}\|_q \leq n^{\frac{1}{q}} \|x - \hat{x}\|_\infty .$$

Therefore

$$\|x^* - \hat{x}\|_\infty - \|x - \hat{x}\|_\infty \leq (n^{\frac{1}{q}} - 1) \|x - \hat{x}\|_\infty \leq (n^{\frac{1}{q}} - 1) 2\rho < 1 ,$$

where the last inequality holds by the choice of q . Since $\|x^* - \hat{x}\|_\infty$ and $\|x - \hat{x}\|_\infty$ are integers we find that $\|x^* - \hat{x}\|_\infty \leq \|x - \hat{x}\|_\infty$. This establishes the claim. \square

In particular, for all positive $p \in \mathbb{Z}_\infty$, using the Graver basis we can solve

$$\min \{ \|x\|_p : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u \} ,$$

which for $p = \infty$ is equivalent to the min-max integer program

$$\min \{ \max\{|x_i| : i = 1, \dots, n\} : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u \} .$$

CONVEX INTEGER MAXIMIZATION. We proceed to discuss the *maximization* of a convex function over of the composite form $f(Wx)$, with $f : \mathbb{Z}^d \rightarrow \mathbb{Z}$ any convex function and W any integer $d \times n$ matrix.

We need a result of [23]. A *linear-optimization oracle* for a set $S \subset \mathbb{Z}^n$ is one that, given $w \in \mathbb{Z}^n$, solves the linear optimization problem $\max\{wx : x \in S\}$. A *direction* of an edge (1-dimensional face) e of a polyhedron P is any nonzero scalar multiple of $u - v$ where u, v are any two distinct points in e . A set of *all edge-directions* of P is one that contains some direction of each edge of P , see Figure 6.

THEOREM 3.4. [23] *For all fixed d there is an algorithm that, given a finite set $S \subset \mathbb{Z}^n$ presented by linear-optimization oracle, integer $d \times n$ matrix W , set $E \subset \mathbb{Z}^n$ of all edge-directions of $\text{conv}(S)$, and convex $f : \mathbb{Z}^d \rightarrow \mathbb{R}$ presented by comparison oracle, solves in time polynomial in $\langle \max\{\|x\|_\infty : x \in S\}, W, E \rangle$, the convex problem*

$$\max \{ f(Wx) : x \in S \} .$$

We now show that, fortunately enough, the Graver basis of a matrix A is a set of all edge-directions of the integer hull related to the integer program defined by A .

LEMMA 3.10. *For every integer $m \times n$ matrix A , $l, u \in \mathbb{Z}_\infty^n$, and $b \in \mathbb{Z}^m$, the Graver basis $\mathcal{G}(A)$ is a set of all edge-directions of $P_I := \text{conv}\{x \in \mathbb{Z}^n : Ax = b, l \leq x \leq u\}$.*

Edge-Directions of a Convex Polytope

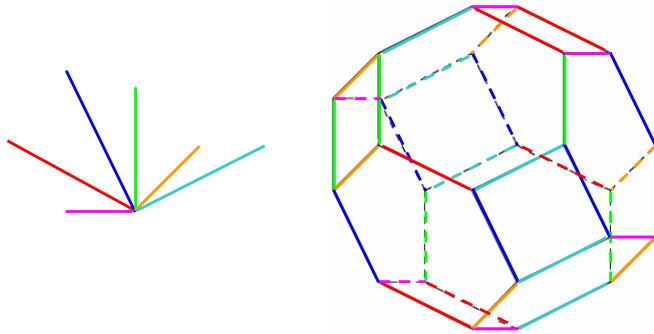


FIG. 6. Edge-directions of a convex polytope.

Proof. Consider any edge e of P_I and pick two distinct integer points $x, y \in e$. Then $g := y - x$ is in $\mathcal{L}^*(A)$ and hence Lemma 3.1 implies that $g = \sum_i h_i$ is a conformal sum for suitable $h_i \in \mathcal{G}(A)$. We claim that $x + h_i \in P_I$ for all i . Indeed, $h_i \in \mathcal{G}(A)$ implies $A(x + h_i) = Ax = b$, and $l \leq x, x + g \leq u$ and $h_i \sqsubseteq g$ imply $l \leq x + h_i \leq u$.

Now let $w \in \mathbb{Z}^n$ be uniquely maximized over P_I at the edge e . Then $wh_i = w(x + h_i) - wx \leq 0$ for all i . But $\sum wh_i = wg = wy - wx = 0$, implying that in fact $wh_i = 0$ and hence $x + h_i \in e$ for all i . This implies that h_i is a direction of e (in fact, all h_i are the same and g is a multiple of some Graver basis element). \square

Using Theorems 3.2 and 3.4 and Lemma 3.10 we obtain the following theorem.

THEOREM 3.5. [5] *For every fixed d there is an algorithm that, given integer $m \times n$ matrix A , its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}_{\infty}^n$, $b \in \mathbb{Z}^m$, integer $d \times n$ matrix W , and convex function $f : \mathbb{Z}^d \rightarrow \mathbb{R}$ presented by a comparison oracle, solves in time which is polynomial in $\langle A, W, \mathcal{G}(A), l, u, b \rangle$, the convex integer maximization problem*

$$\max \{f(Wx) : x \in \mathbb{Z}^n, Ax = b, l \leq x \leq u\} .$$

Proof. Let $S := \{x \in \mathbb{Z}^n : Ax = b, l \leq x \leq u\}$. The algorithm of Theorem 3.2 allows to simulate in polynomial time a linear-optimization oracle for S . In particular, it allows to either conclude that S is infinite and stop or conclude that it is finite, in which case $\langle \max\{\|x\|_\infty : x \in S\} \rangle$ is polynomial in $\langle A, l, u, b \rangle$, and continue. By Lemma 3.10, the given Graver basis is a set of all edge-directions of $\text{conv}(S) = P_I$. Hence the algorithm of Theorem 3.4 can be applied, and provides the polynomial time solution of the convex integer maximization program. \square

3.2. N-fold integer programming. In this subsection we focus our attention on (nonlinear) n -fold integer programming. In §3.2.1 we study Graver bases of n -fold products of integer bimatrices and show that they can be computed in polynomial time. In §3.2.2 we combine the results of §3.1 and §3.2.1, and prove our Theorems 1.1–1.5, which establish the polynomial time solvability of linear and nonlinear n -fold integer programming.

3.2.1. Graver bases of n-fold products. Let A be a fixed integer $(r, s) \times t$ bimatrix with blocks A_1, A_2 . For each positive integer n we index vectors in \mathbb{Z}^{nt} as $x = (x^1, \dots, x^n)$ with each *brick* x^k lying in \mathbb{Z}^t . The *type* of vector x is the number $\text{type}(x) := |\{k : x^k \neq 0\}|$ of nonzero bricks of x .

The following definition plays an important role in the sequel.

DEFINITION 3.2. [24] The *Graver complexity* of an integer bimatrix A is defined as

$$g(A) := \inf \{g \in \mathbb{Z}_+ : \text{type}(x) \leq g \text{ for all } x \in \mathcal{G}(A^{(n)}) \text{ and all } n\}.$$

We proceed to establish a result of [24] and its extension in [16] which show that, in fact, the Graver complexity of every integer bimatrix A is finite.

Consider n -fold products $A^{(n)}$ of A . By definition of the n -fold product, $A^{(n)}x = 0$ if and only if $A_1 \sum_{k=1}^n x^k = 0$ and $A_2 x^k = 0$ for all k . In particular, a necessary condition for x to lie in $\mathcal{L}(A^{(n)})$, and in particular in $\mathcal{G}(A^{(n)})$, is that $x^k \in \mathcal{L}(A_2)$ for all k . Call a vector $x = (x^1, \dots, x^n)$ *full* if, in fact, $x^k \in \mathcal{L}^*(A_2)$ for all k , in which case $\text{type}(x) = n$, and *pure* if, moreover, $x^k \in \mathcal{G}(A_2)$ for all k . Full vectors, and in particular pure vectors, are natural candidates for lying in the Graver basis $\mathcal{G}(A^{(n)})$ of $A^{(n)}$, and will indeed play an important role in its construction.

Consider any full vector $y = (y^1, \dots, y^m)$. By definition, each brick of y satisfies $y^i \in \mathcal{L}^*(A_2)$ and is therefore a conformal sum $y^i = \sum_{j=1}^{k_i} x^{i,j}$ of some elements $x^{i,j} \in \mathcal{G}(A_2)$ for all i, j . Let $n := k_1 + \dots + k_m \geq m$ and let x be the pure vector

$$x = (x^1, \dots, x^n) := (x^{1,1}, \dots, x^{1,k_1}, \dots, x^{m,1}, \dots, x^{m,k_m}).$$

We call the pure vector x an *expansion* of the full vector y , and we call the full vector y a *compression* of the pure vector x . Note that $A_1 \sum y^i = A_1 \sum x^{i,j}$ and therefore $y \in \mathcal{L}(A^{(m)})$ if and only if $x \in \mathcal{L}(A^{(n)})$. Note also

that each full y may have many different expansions and each pure x may have many different compressions.

LEMMA 3.11. *Consider any full $y = (y^1, \dots, y^m)$ and any expansion $x = (x^1, \dots, x^n)$ of y . If y is in the Graver basis $\mathcal{G}(A^{(m)})$ then x is in the Graver basis $\mathcal{G}(A^{(n)})$.*

Proof. Let $x = (x^{1,1}, \dots, x^{m,k_m}) = (x^1, \dots, x^n)$ be an expansion of $y = (y^1, \dots, y^m)$ with $y^i = \sum_{j=1}^{k_i} x^{i,j}$ for each i . Suppose indirectly $y \in \mathcal{G}(A^{(m)})$ but $x \notin \mathcal{G}(A^{(n)})$. Since $y \in \mathcal{L}^*(A^{(m)})$ we have $x \in \mathcal{L}^*(A^{(n)})$. Since $x \notin \mathcal{G}(A^{(n)})$, there exists an element $g = (g^{1,1}, \dots, g^{m,k_m})$ in $\mathcal{G}(A^{(n)})$ satisfying $g \sqsubset x$. Let $h = (h^1, \dots, h^m)$ be the compression of g defined by $h^i := \sum_{j=1}^{k_i} g^{i,j}$. Since $g \in \mathcal{L}^*(A^{(n)})$ we have $h \in \mathcal{L}^*(A^{(m)})$. But $h \sqsubset y$, contradicting $y \in \mathcal{G}(A^{(m)})$. This completes the proof. \square

LEMMA 3.12. *The Graver complexity $g(A)$ of every integer bimatrix A is finite.*

Proof. We need to bound the type of any element in the Graver basis of the l -fold product of A for any l . Suppose there is an element z of type m in some $\mathcal{G}(A^{(l)})$. Then its restriction $y = (y^1, \dots, y^m)$ to its m nonzero bricks is a full vector and is in the Graver basis $\mathcal{G}(A^{(m)})$. Let $x = (x^1, \dots, x^n)$ be any expansion of y . Then $\text{type}(z) = m \leq n = \text{type}(x)$, and by Lemma 3.11, the pure vector x is in $\mathcal{G}(A^{(n)})$.

Therefore, it suffices to bound the type of any pure element in the Graver basis of the n -fold product of A for any n . Suppose $x = (x^1, \dots, x^n)$ is a pure element in $\mathcal{G}(A^{(n)})$ for some n . Let $\mathcal{G}(A_2) = \{g^1, \dots, g^p\}$ be the Graver basis of A_2 and let G_2 be the $t \times p$ matrix whose columns are the g^i . Let $v \in \mathbb{Z}_+^p$ be the vector with $v_i := |\{k : x^k = g^i\}|$ counting the number of bricks of x which are equal to g^i for each i . Then $\sum_{i=1}^p v_i = \text{type}(x) = n$. Now, note that $A_1 G_2 v = A_1 \sum_{k=1}^n x^k = 0$ and hence $v \in \mathcal{L}^*(A_1 G_2)$. We claim that, moreover, v is in $\mathcal{G}(A_1 G_2)$. Suppose indirectly it is not. Then there is a $\hat{v} \in \mathcal{G}(A_1 G_2)$ with $\hat{v} \sqsubset v$, and it is easy to obtain a nonzero $\hat{x} \sqsubset x$ from x by zeroing out some bricks so that $\hat{v}_i = |\{k : \hat{x}^k = g^i\}|$ for all i . Then $A_1 \sum_{k=1}^n \hat{x}^k = A_1 G_2 \hat{v} = 0$ and hence $\hat{x} \in \mathcal{L}^*(A^{(n)})$, contradicting $x \in \mathcal{G}(A^{(n)})$.

So the type of any pure vector, and hence the Graver complexity of A , is at most the largest value $\sum_{i=1}^p v_i$ of any nonnegative vector v in the Graver basis $\mathcal{G}(A_1 G_2)$. \square

We proceed to establish the following theorem from [4] which asserts that Graver bases of n -fold products can be computed in polynomial time. An n -lifting of a vector $y = (y^1, \dots, y^m)$ consisting of m bricks is any vector $z = (z^1, \dots, z^n)$ consisting of n bricks such that for some $1 \leq k_1 < \dots < k_m \leq n$ we have $z^{k_i} = y^i$ for $i = 1, \dots, m$, and all other bricks of z are zero; in particular, $n \geq m$ and $\text{type}(z) = \text{type}(y)$.

THEOREM 3.6. [4] *For every fixed integer bimatrix A there is an algorithm that, given positive integer n , computes the Graver basis $\mathcal{G}(A^{(n)})$ of the n -fold product of A , in time which is polynomial in n . In particular,*

the cardinality $|\mathcal{G}(A^{(n)})|$ and the binary-encoding length $\langle \mathcal{G}(A^{(n)}) \rangle$ of the Graver basis of $A^{(n)}$ are polynomial in n .

Proof. Let $g := g(A)$ be the Graver complexity of A . Since A is fixed, so is g . Therefore, for every $n \leq g$, the Graver basis $\mathcal{G}(A^{(n)})$, and in particular, the Graver basis $\mathcal{G}(A^{(g)})$ of the g -fold product of A , can be computed in constant time.

Now, consider any $n > g$. We claim that $\mathcal{G}(A^{(n)})$ satisfies

$$\mathcal{G}(A^{(n)}) = \left\{ z : z \text{ is an } n\text{-lifting of some } y \in \mathcal{G}(A^{(g)}) \right\} .$$

Consider any n -lifting z of any $y \in \mathcal{G}(A^{(g)})$. Suppose indirectly $z \notin \mathcal{G}(A^{(n)})$. Then there exists $z' \in \mathcal{G}(A^{(n)})$ with $z' \sqsubset z$. But then z' is the n -lifting of some $y' \in \mathcal{L}^*(A^{(g)})$ with $y' \sqsubset y$, contradicting $y \in \mathcal{G}(A^{(g)})$. So $z \in \mathcal{G}(A^{(n)})$.

Conversely, consider any $z \in \mathcal{G}(A^{(n)})$. Then $\text{type}(z) \leq g$ and hence z is the n -lifting of some $y \in \mathcal{L}^*(A^{(g)})$. Suppose indirectly $y \notin \mathcal{G}(A^{(g)})$. Then there exists $y' \in \mathcal{G}(A^{(g)})$ with $y' \sqsubset y$. But then the n -lifting z' of y' satisfies $z' \in \mathcal{L}^*(A^{(n)})$ with $z' \sqsubset z$, contradicting $z \in \mathcal{G}(A^{(n)})$. So $y \in \mathcal{G}(A^{(g)})$.

Now, the number of n -liftings of each $y \in \mathcal{G}(A^{(g)})$ is at most $\binom{n}{g}$, and hence

$$|\mathcal{G}(A^{(n)})| \leq \binom{n}{g} |\mathcal{G}(A^{(g)})| = O(n^g) .$$

So the set of all n -liftings of vectors in $\mathcal{G}(A^{(g)})$ and hence the Graver basis $\mathcal{G}(A^{(n)})$ of the n -fold product can be computed in time polynomial in n as claimed. \square

3.2.2. N-fold integer programming in polynomial time. Combining Theorem 3.6 and the results of §3.1 we now obtain Theorems 1.1–1.4.

Theorem 1.1 [4] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given positive integer $n, l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and $w \in \mathbb{Z}^{nt}$, solves in time which is polynomial in n and $\langle l, u, b, w \rangle$, the following linear n -fold integer program,*

$$\min \left\{ wx : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u \right\} .$$

Proof. Compute the Graver basis $\mathcal{G}(A^{(n)})$ using the algorithm of Theorem 3.6. Now apply the algorithm of Theorem 3.2 with this Graver basis and solve the problem. \square

Theorem 1.2 [12] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given $n, l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and separable convex $f : \mathbb{Z}^{nt} \rightarrow \mathbb{Z}$ presented by a comparison oracle, solves in time polynomial in n and $\langle l, u, b, f \rangle$, the program*

$$\min \left\{ f(x) : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u \right\} .$$

Proof. Compute the Graver basis $\mathcal{G}(A^{(n)})$ using the algorithm of Theorem 3.6. Now apply the algorithm of Theorem 3.1 with this Graver basis and solve the problem. \square

Theorem 1.3 [12] *For each fixed integer $(r, s) \times t$ bimatrix A , there is an algorithm that, given positive integers n and $p, l, u \in \mathbb{Z}_{\infty}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and $\hat{x} \in \mathbb{Z}^{nt}$, solves in time polynomial in n , p , and $\langle l, u, b, \hat{x} \rangle$, the following distance minimization program,*

$$\min \{ \|x - \hat{x}\|_p : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u \}. \quad (3.10)$$

For $p = \infty$ the problem (3.10) can be solved in time polynomial in n and $\langle l, u, b, \hat{x} \rangle$.

Proof. Compute the Graver basis $\mathcal{G}(A^{(n)})$ using the algorithm of Theorem 3.6. Now apply the algorithm of Theorem 3.3 with this Graver basis and solve the problem. \square

Theorem 1.4 [5] *For each fixed d and $(r, s) \times t$ integer bimatrix A , there is an algorithm that, given n , bounds $l, u \in \mathbb{Z}_{\infty}^{nt}$, integer $d \times nt$ matrix W , $b \in \mathbb{Z}^{r+ns}$, and convex function $f : \mathbb{Z}^d \rightarrow \mathbb{R}$ presented by a comparison oracle, solves in time polynomial in n and $\langle W, l, u, b \rangle$, the convex n -fold integer maximization program*

$$\max \{f(Wx) : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, l \leq x \leq u\}.$$

Proof. Compute the Graver basis $\mathcal{G}(A^{(n)})$ using the algorithm of Theorem 3.6. Now apply the algorithm of Theorem 3.5 with this Graver basis and solve the problem. \square

3.2.3. Weighted separable convex integer minimization. We proceed to establish Theorem 1.5 which is a broad extension of Theorem 1.2 that allows the objective function to include a composite term of the form $f(Wx)$, where $f : \mathbb{Z}^d \rightarrow \mathbb{Z}$ is a separable convex function and W is an integer matrix with d rows, and to incorporate inequalities on Wx . We begin with two lemmas. As before, \hat{f}, \hat{g} denote the maximum values of $|f(Wx)|, |g(x)|$ over the feasible set.

LEMMA 3.13. *There is an algorithm that, given an integer $m \times n$ matrix A , an integer $d \times n$ matrix W , $l, u \in \mathbb{Z}_{\infty}^n$, $\hat{l}, \hat{u} \in \mathbb{Z}_{\infty}^d$, $b \in \mathbb{Z}^m$, the Graver basis $\mathcal{G}(B)$ of*

$$B := \begin{pmatrix} A & 0 \\ W & I \end{pmatrix},$$

and separable convex functions $f : \mathbb{Z}^d \rightarrow \mathbb{Z}$, $g : \mathbb{Z}^n \rightarrow \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial in $\langle A, W, \mathcal{G}(B), l, u, \hat{l}, \hat{u}, b, \hat{f}, \hat{g} \rangle$, the problem

$$\min \{f(Wx) + g(x) : x \in \mathbb{Z}^n, Ax = b, \hat{l} \leq Wx \leq \hat{u}, l \leq x \leq u\}. \quad (3.11)$$

Proof. Define $h : \mathbb{Z}^{n+d} \rightarrow \mathbb{Z}$ by $h(x, y) := f(-y) + g(x)$ for all $x \in \mathbb{Z}^n$ and $y \in \mathbb{Z}^d$. Clearly, h is separable convex since f, g are. Now, problem (3.11) can be rewritten as

$$\min\{h(x, y) : (x, y) \in \mathbb{Z}^{n+d}\ , \\ \left(\begin{array}{cc} A & 0 \\ W & I \end{array}\right) \left(\begin{array}{c} x \\ y \end{array}\right) = \left(\begin{array}{c} b \\ 0 \end{array}\right), l \leq x \leq u, -\hat{u} \leq y \leq -\hat{l}\}$$

and the statement follows at once by applying Theorem 3.1 to this problem. \square

LEMMA 3.14. *For every fixed integer $(r, s) \times t$ bimatrix A and $(p, q) \times t$ bimatrix W , there is an algorithm that, given any positive integer n , computes in time polynomial in n , the Graver basis $\mathcal{G}(B)$ of the following $(r + ns + p + nq) \times (nt + p + nq)$ matrix,*

$$B := \left(\begin{array}{cc} A^{(n)} & 0 \\ W^{(n)} & I \end{array}\right).$$

Proof. Let D be the $(r + p, s + q) \times (t + p + q)$ bimatrix whose blocks are defined by

$$D_1 := \left(\begin{array}{ccc} A_1 & 0 & 0 \\ W_1 & I_p & 0 \end{array}\right), \quad D_2 := \left(\begin{array}{ccc} A_2 & 0 & 0 \\ W_2 & 0 & I_q \end{array}\right).$$

Apply the algorithm of Theorem 3.6 and compute in polynomial time the Graver basis $\mathcal{G}(D^{(n)})$ of the n -fold product of D , which is the following matrix:

$$D^{(n)} = \left(\begin{array}{ccc|ccc|c|ccc} A_1 & 0 & 0 & A_1 & 0 & 0 & \cdots & A_1 & 0 & 0 \\ W_1 & I_p & 0 & W_1 & I_p & 0 & \cdots & W_1 & I_p & 0 \\ \hline A_2 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ W_2 & 0 & I_q & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & A_2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & W_2 & 0 & I_q & \cdots & 0 & 0 & 0 \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & W_2 & 0 & I_q \end{array}\right).$$

Suitable row and column permutations applied to $D^{(n)}$ give the following matrix:

$$C := \left(\begin{array}{cccc|cccc|cccc} A_1 & A_1 & \cdots & A_1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ A_2 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_2 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \hline W_1 & W_1 & \cdots & W_1 & I_p & I_p & \cdots & I_p & 0 & 0 & \cdots & 0 \\ W_2 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & I_q & 0 & \cdots & 0 \\ 0 & W_2 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & I_q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W_2 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & I_q \end{array} \right).$$

Obtain the Graver basis $\mathcal{G}(C)$ in polynomial time from $\mathcal{G}(D^{(n)})$ by permuting the entries of each element of the latter by the permutation of the columns of $\mathcal{G}(D^{(n)})$ that is used to get C (the permutation of the rows does not affect the Graver basis).

Now, note that the matrix B can be obtained from C by dropping all but the first p columns in the second block. Consider any element in $\mathcal{G}(C)$, indexed, according to the block structure, as

$$(x^1, x^2, \dots, x^n, y^1, y^2, \dots, y^n, z^1, z^2, \dots, z^n).$$

Clearly, if $y^k = 0$ for $k = 2, \dots, n$ then the restriction

$$(x^1, x^2, \dots, x^n, y^1, z^1, z^2, \dots, z^n)$$

of this element is in the Graver basis of B . On the other hand, if

$$(x^1, x^2, \dots, x^n, y^1, z^1, z^2, \dots, z^n)$$

is any element in $\mathcal{G}(B)$ then its extension

$$(x^1, x^2, \dots, x^n, y^1, 0, \dots, 0, z^1, z^2, \dots, z^n)$$

is clearly in $\mathcal{G}(C)$. So the Graver basis of B can be obtained in polynomial time by

$$\begin{aligned} \mathcal{G}(B) := \{(x^1, \dots, x^n, y^1, z^1, \dots, z^n) : \\ (x^1, \dots, x^n, y^1, 0, \dots, 0, z^1, \dots, z^n) \in \mathcal{G}(C)\}. \end{aligned}$$

This completes the proof. \square

Theorem 1.5 [13] *For each fixed integer $(r, s) \times t$ bimatrix A and integer $(p, q) \times t$ bimatrix W , there is an algorithm that, given $n, l, u \in \mathbb{Z}_{\infty}^{nt}, \hat{l}, \hat{u} \in \mathbb{Z}_{\infty}^{p+nq}$, $b \in \mathbb{Z}^{r+ns}$, and separable convex functions $f : \mathbb{Z}^{p+nq} \rightarrow \mathbb{Z}$, $g :$*

$\mathbb{Z}^{nt} \rightarrow \mathbb{Z}$ presented by evaluation oracles, solves in time polynomial in n and $\langle l, u, \hat{l}, \hat{u}, b, \hat{f}, \hat{g} \rangle$, the generalized program

$$\min\{f(W^{(n)}x) + g(x) : x \in \mathbb{Z}^{nt}, A^{(n)}x = b, \hat{l} \leq W^{(n)}x \leq \hat{u}, l \leq x \leq u\}.$$

Proof. Use the algorithm of Lemma 3.14 to compute the Graver basis $\mathcal{G}(B)$ of

$$B := \begin{pmatrix} A^{(n)} & 0 \\ W^{(n)} & I \end{pmatrix}.$$

Now apply the algorithm of Lemma 3.13 and solve the nonlinear integer program. \square

4. Discussion. We conclude with a short discussion of the universality of n -fold integer programming and the Graver complexity of (directed) graphs, a new important invariant which controls the complexity of our multiway table and multicommodity flow applications.

4.1. Universality of n -fold integer programming. Let us introduce the following notation. For an integer $s \times t$ matrix D , let $\boxplus D$ denote the $(t, s) \times t$ bimatrix whose first block is the $t \times t$ identity matrix and whose second block is D . Consider the following special form of the n -fold product, defined for a matrix D , by $D^{[n]} := (\boxplus D)^{(n)}$. We consider such m -fold products of the 1×3 matrix $\mathbf{1}_3 := [1, 1, 1]$. Note that $\mathbf{1}_3^{[m]}$ is precisely the $(3+m) \times 3m$ incidence matrix of the complete bipartite graph $K_{3,m}$. For instance, for $m = 3$, it is the matrix

$$\mathbf{1}_3^{[3]} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

We can now rewrite Theorem 2.1 in the following compact and elegant form.

The Universality Theorem [7] *Every rational polytope $\{y \in \mathbb{R}_+^d : Ay = b\}$ stands in polynomial time computable integer preserving bijection with some polytope*

$$\left\{ x \in \mathbb{R}_+^{3mn} : \mathbf{1}_3^{[m][n]}x = a \right\}. \quad (4.1)$$

The bijection constructed by the algorithm of this theorem is, moreover, a simple projection from \mathbb{R}^{3mn} to \mathbb{R}^d that erases all but some d coordinates (see [7]). For $i = 1, \dots, d$ let $x_{\sigma(i)}$ be the coordinate of x that is mapped to y_i under this projection. Then any linear or nonlinear integer program

$\min\{f(y) : y \in \mathbb{Z}_+^d, Ay = b\}$ can be lifted in polynomial time to the following integer program over a simple $\{0, 1\}$ -valued matrix $\mathbf{1}_3^{[m][n]}$ which is completely determined by two parameters m and n only,

$$\min \left\{ f(x_{\sigma(1)}, \dots, x_{\sigma(d)}) : x \in \mathbb{Z}_+^{3mn}, \mathbf{1}_3^{[m][n]}x = a \right\}. \quad (4.2)$$

This also shows the universality of n -fold integer programming: every linear or nonlinear integer program is equivalent to an n -fold integer program over some bimatrix $\boxminus \mathbf{1}_3^{[m]}$ which is completely determined by a single parameter m .

Moreover, for every fixed m , program (4.2) can be solved in polynomial time for linear forms and broad classes of convex and concave functions by Theorems 1.1–1.5.

4.2. Graver complexity of graphs and digraphs. The significance of the following new (di)-graph invariant will be explained below.

DEFINITION 4.1. [1] The *Graver complexity* of a graph or a digraph G is the Graver complexity $g(G) := g(\boxminus D)$ of the bimatrix $\boxminus D$ with D the incidence matrix of G .

One major task done by our algorithms for linear and nonlinear n -fold integer programming over a bimatrix A is the construction of the Graver basis $\mathcal{G}(A^{(n)})$ in time $O(n^{g(A)})$ with $g(A)$ the Graver complexity of A (see proof of Theorem 3.6).

Since the bimatrix underlying the universal n -fold integer program (4.2) is precisely $\boxminus D$ with $D = \mathbf{1}_3^{[m]}$ the incidence matrix of $K_{3,m}$, it follows that the complexity of computing the relevant Graver bases for this program for fixed m and variable n is $O(n^{g(K_{3,m})})$ where $g(K_{3,m})$ is the Graver complexity of $K_{3,m}$ as just defined.

Turning to the many-commodity transshipment problem over a digraph G discussed in §2.2.1, the bimatrix underlying the n -fold integer program (2.1) in the proof of Theorem 2.5 is precisely $\boxminus D$ with D the incidence matrix of G , and so it follows that the complexity of computing the relevant Graver bases for this program is $O(n^{g(G)})$ where $g(G)$ is the Graver complexity of the digraph G as just defined.

So the Graver complexity of a (di)-graph controls the complexity of computing the Graver bases of the relevant n -fold integer programs, and hence its significance.

Unfortunately, our present understanding of the Graver complexity of (di)-graphs is very limited and much more study is required. Very little is known even for the complete bipartite graphs $K_{3,m}$: while $g(K_{3,3}) = 9$, already $g(K_{3,4})$ is unknown. See [1] for more details and a lower bound on $g(K_{3,m})$ which is exponential in m .

Acknowledgements. I thank Jon Lee and Sven Leyffer for inviting me to write this article. I am indebted to Jesus De Loera, Raymond Hemmecke, Uriel Rothblum and Robert Weismantel for their collaboration in

developing the theory of n -fold integer programming, and to Raymond Hemmecke for his invaluable suggestions. The article was written mostly while I was visiting and delivering the Nachdiplom Lectures at ETH Zürich during Spring 2009. I thank the following colleagues at ETH for useful feedback: David Adjiashvili, Jan Foniok, Martin Fuchsberger, Komei Fukuda, Dan Hefetz, Hans-Rudolf Künsch, Hans-Jakob Lüthi, and Philipp Zumstein. I also thank Dorit Hochbaum, Peter Malkin, and a referee for useful remarks.

REFERENCES

- [1] BERSTEIN Y. AND ONN S., *The Graver complexity of integer programming*, Annals Combin. **13** (2009) 289–296.
- [2] COOK W., FONLUPT J., AND SCHRIJVER A., *An integer analogue of Carathéodory's theorem*, J. Comb. Theory Ser. B **40** (1986) 63–70.
- [3] COX L.H., *On properties of multi-dimensional statistical tables*, J. Stat. Plan. Infer. **117** (2003) 251–273.
- [4] DE LOERA J., HEMMECKE R., ONN S., AND WEISMANTEL R., *N-fold integer programming*, Disc. Optim. **5** (Volume in memory of George B. Dantzig) (2008) 231–241.
- [5] DE LOERA J., HEMMECKE R., ONN S., ROTHBLUM U.G., AND WEISMANTEL R., *Convex integer maximization via Graver bases*, J. Pure App. Algeb. **213** (2009) 1569–1577.
- [6] DE LOERA J. AND ONN S., *The complexity of three-way statistical tables*, SIAM J. Comp. **33** (2004) 819–836.
- [7] DE LOERA J. AND ONN S., *All rational polytopes are transportation polytopes and all polytopal integer sets are contingency tables*, In: Proc. IPCO 10 – Symp. on Integer Programming and Combinatorial Optimization (Columbia University, New York), Lec. Not. Comp. Sci., Springer **3064** (2004) 338–351.
- [8] DE LOERA J. AND ONN S., *Markov bases of three-way tables are arbitrarily complicated*, J. Symb. Comp. **41** (2006) 173–181.
- [9] FIENBERG S.E. AND RINALDO A., *Three centuries of categorical data analysis: Log-linear models and maximum likelihood estimation*, J. Stat. Plan. Infer. **137** (2007) 3430–3445.
- [10] GORDAN P., *Über die Auflösung linearer Gleichungen mit reellen Coefficienten*, Math. Annalen **6** (1873) 23–28.
- [11] GRAVER J.E., *On the foundations of linear and linear integer programming I*, Math. Prog. **9** (1975) 207–226.
- [12] HEMMECKE R., ONN S., AND WEISMANTEL R., *A polynomial oracle-time algorithm for convex integer minimization*, Math. Prog. **126** (2011) 97–117.
- [13] HEMMECKE R., ONN S., AND WEISMANTEL R., *N-fold integer programming and nonlinear multi-transshipment*, Optimization Letters **5** (2011) 13–25.
- [14] HOCHBAUM D.S. AND SHANTHIKUMAR J.G., *Convex separable optimization is not much harder than linear optimization*, J. Assoc. Comp. Mach. **37** (1990) 843–862.
- [15] HOFFMAN A.J. AND KRUSKAL J.B., *Integral boundary points of convex polyhedra*, In: Linear inequalities and Related Systems, Ann. Math. Stud. **38**, 223–246, Princeton University Press, Princeton, NJ (1956).
- [16] HOŞTEN S. AND SULLIVANT S., *Finiteness theorems for Markov bases of hierarchical models*, J. Comb. Theory Ser. A **114** (2007) 311–321.
- [17] IRVING R. AND JERRUM M.R., *Three-dimensional statistical data security problems*, SIAM J. Comp. **23** (1994) 170–184.

- [18] LENSTRA JR. H.W., *Integer programming with a fixed number of variables*, Math. Oper. Res. **8** (1983) 538–548.
- [19] MOTZKIN T.S., *The multi-index transportation problem*, Bull. Amer. Math. Soc. **58** (1952) 494.
- [20] ONN S., *Entry uniqueness in margined tables*, In: Proc. PSD 2006 – Symp. on Privacy in Statistical Databases (Rome, Italy), Lec. Not. Comp. Sci., Springer **4302** (2006) 94–101.
- [21] ONN S., *Convex discrete optimization*, In: Encyclopedia of Optimization, Springer (2009) 513–550.
- [22] ONN S., *Nonlinear discrete optimization*, Zurich Lectures in Advanced Mathematics, European Mathematical Society, 2010.
- [23] ONN S. AND ROTHBLUM U.G., *Convex combinatorial optimization*, Disc. Comp. Geom. **32** (2004) 549–566.
- [24] SANTOS F. AND STURMFELS B., *Higher Lawrence configurations*, J. Comb. Theory Ser. A **103** (2003) 151–164.
- [25] SCHRIJVER A., *Theory of Linear and Integer Programming*, Wiley, New York (1986).
- [26] SEBÖ, A., *Hilbert bases, Carathéodory's theorem and combinatorial optimization*, In: Proc. IPCO 1 - 1st Conference on Integer Programming and Combinatorial Optimization (R. Kannan and W.R. Pulleyblank Eds.) (1990) 431–455.
- [27] VLACH M., *Conditions for the existence of solutions of the three-dimensional planar transportation problem*, Disc. App. Math. **13** (1986) 61–78.
- [28] YEMELICHEV V.A., KOVALEV M.M., AND KRAVTSOV M.K., *Polytopes, Graphs and Optimisation*, Cambridge University Press, Cambridge (1984).

PART IX:

APPLICATIONS

MINLP APPLICATION FOR ACH INTERIORS RESTRUCTURING

ERICA KLAMPFL* AND YAKOV FRADKIN*

Abstract. In 2006, Ford Motor Company committed to restructure the \$1.5 billion ACH interiors business. This extensive undertaking required a complete re-engineering of the supply footprint of 42 high-volume product lines over 26 major manufacturing processes and more than 50 potential supplier sites. To enable data-driven decision making, we developed a decision support system (DSS) that could quickly yield a variety of solutions for different business scenarios. To drive this DSS, we developed a non-standard mathematical model for the assignment problem and a novel practical approach to solve the resulting large-scale mixed-integer nonlinear program (MINLP). In this paper, we present the MINLP and describe how we reformulated it to remove the nonlinearity in the objective function, while still capturing the supplier facility cost as a function of the supplier's utilization. We also describe our algorithm and decoupling method that scale well with the problem size and avoid the nonlinearity in the constraints. Finally, we provide a computational example to demonstrate the algorithm's functionality.

Key words. Mixed-integer programming, applications of mathematical programming.

AMS(MOS) subject classifications. Primary 90C11, 90C90.

1. Introduction. As part of the 2006 Accelerated Way Forward plan, Ford committed to sell or close all existing Automotive Components Holdings, LLC (ACH) plants by the end of 2008 [4]. In this paper, we focus on the interiors business portion of the ACH business, which was a major piece of the ACH portfolio, about \$1.5 billion in annual revenue. Ford's proactive Way Forward approach was key in the ACH interiors business' avoidance of bankruptcy to which many other interiors suppliers (e.g. Plastech [10], Blue Water [11]) succumbed.

Making strategic sourcing decisions is well suited to operations research techniques, as there are often millions of possible alternatives that cannot be readily evaluated or analyzed manually. We developed a quantitative decision support system that allowed the major stakeholders to assign the interiors product programs to suppliers that had available capacity at their facilities.

The resulting mathematical program that describes this sourcing problem is a large-scale mixed-integer nonlinear program (MINLP); the nonlinearity is due to a product program's production cost being a function of a facility's capacity utilization (suppliers prefer to operate at almost full capacity to get the full value of the fixed cost of their equipment and facility). Since it is not well known how to solve MINLPs of this size, we developed a MINLP reformulation that removed the nonlinearity in the

*Ford Research & Advanced Engineering, Systems Analytics & Environmental Sciences Department, Dearborn, MI 48124.

objective function, and subsequently, we developed an algorithm based on decoupling the reformulated MINLP into solvable mixed-integer programs (MIPs).

In this paper, we first provide an overview of the business problem in Section 2. Next, in Section 3, we describe the MINLP formulation together with the constraints that provide details on the business needs, such as certain products being constrained to be produced together due to shipping requirements or intellectual property licensing. In Section 4, we describe the problem reformulation that removes the nonlinearity in the objective function. The resulting formulation is also a MINLP with the nonlinearity in the constraints remaining. In Section 5, we demonstrate how we decouple this MINLP into two MIPs and describe the algorithm that iteratively solves the MIPs until convergence. In Section 6, we provide a small computational example to demonstrate the solution methodology. Finally, in Section 7, we provide a summary of the business outcome of this work, as well as the MINLP modeling insight that we gained.

2. Problem overview. In this section, we will give a brief overview of the business problem of making the strategic sourcing decisions associated with the ACH restructuring. For details of the business climate, problem, and scenario analysis see [7].

ACH's interiors business was located in two large-scale manufacturing plants in southeast Michigan (Utica and Saline), producing instrument panels, consoles, door trim panels and cockpit module assemblies for Ford assembly plants. Saline and Utica were underutilized and not profitable, and the interiors business was consuming a substantial amount of company cash. Ford had committed to sell or close all existing ACH plants, including Utica and Saline, by the end of 2008 as a part of the 2006 Accelerated Way Forward plan. However, after traditional improvement actions were made and a concerted marketing effort was attempted in 2006, Ford was unable to find an interested, qualified buyer for the interiors business.

ACH and Ford Purchasing were left with a few options: consolidate the product programs into either Saline or Utica, making the consolidated plant more attractive to a potential investor due to better facility utilization; outsource all product programs to suppliers that have facilities with available capacity; or devise some combination of restructuring and outsourcing. How to make this decision became daunting, as there was an overwhelming number of possible sourcing alternatives of the interiors business to the overall supply base: more than 40 product programs requiring multiple manufacturing processes that could be sourced to over 50 potential production sites, each with different configurations of production technologies, capacities, utilizations and shipping distances to assembly plants.

It soon became apparent that the quality of these decisions could be drastically improved if a computational tool was available for identifying

and evaluating the millions of possible product programs-to-suppliers allocations. Specifically, the business problem centered on identifying the best possible sourcing pattern for the product programs at the two ACH interiors plants, given certain physical, financial and operational considerations. In this case, the “best possible sourcing pattern” was defined as a sourcing footprint that yielded the minimum net present value of five-year cost stream to Ford, factoring in 1) investment costs to achieve the new sourcing footprint, 2) the resulting annual purchased part expenditure (dependent on facility utilization), and 3) the annual freight expenditure to ship the interiors products to their respective assembly destinations.

ACH and Ford Purchasing wanted a model that faithfully represented the business problem by accounting for the nonlinear supplier facility cost curve as a function of capacity utilization, as well as the many business constraints. They also wanted a means for allowing ACH customers the ability to independently run the tool and perform scenario analysis.

3. MINLP formulation. The underlying model that represents the interiors restructuring problem is a large-scale MINLP. The MINLP field is still an active area of research (see [5] for recent developments), with no general methods available for large-scale problems: most early work was done in the eighties for problems arising in chemical engineering [1]. The nonlinearity in our MINLP is due to the product program’s production cost being a function of a facility’s capacity utilization. The standard MIP modeling approach that uses a fixed cost plus per-unit variable cost as described in [8] was not readily applicable here, as Ford does not directly pay a third party supplier’s fixed cost. However, Ford is likely to get a reduced per unit cost from the third party supplier dependent upon the facility’s total utilization. This potential reduction in per unit total cost is what we attempt to capture in our modeling approach.

3.1. Input information. The list below describes the inputs and calculated coefficients that we use in the model.

\mathcal{S}	$= \{1, \dots, m\}$ set of facilities with available processes; includes both Ford-owned and third-party facilities.
$\mathcal{S}^c \subseteq \mathcal{S}$	set of excluded (closed-for-business) facilities.
\mathcal{C}	$= \{1, \dots, n\}$ set of manufacturing requirements: a product program may have multiple such requirements.
$\mathcal{C}^p \subseteq \mathcal{C}$	set of indices with cardinality λ that corresponds to unique product programs.
\mathcal{P}	$= \{1, \dots, r\}$ set of processes.
NPV	= net present value multiplier used to discount future annual costs in order to properly reflect the time value of money.
COF	= capacity overload factor: allows override of a facility’s capacity.
$UMIN$	= fewest number of employees required across all union facilities.
CC_j	= cost of closing a Ford-owned facility $j \in \mathcal{S}^c$.

U_j	= designates if a facility $j \in \mathcal{S}$ is a union facility: $U_j = 1$ if facility j is a union facility; otherwise $U_j = 0$.
SU_j	= supplier to which a facility $j \in \mathcal{S}$ belongs.
EH_i	= employees needed to produce product program $i \in \mathcal{C}^p$.
IF_i	= intellectual property family for product program $i \in \mathcal{C}^p$. $IF_i = 0$ if product program i does not fall under any intellectual property agreement. The products in the same intellectual property family must be assigned to the same supplier but possibly to different facilities of that supplier.
VV_i	= production volume for product program $i \in \mathcal{C}^p$.
CPM_i	= product program $i \in \mathcal{C}^p$ freight-family identifier. If the value is greater than zero, it provides the group identifier for product programs that can be shipped for one combined cost.
PF_i	= product-program family-number identifier for each requirement $i \in \mathcal{C}$ that is used to ensure that all requirements of the same product program are assigned to the same facility.
PR_i	= process quantity required for requirement $i \in \mathcal{C}$.
FC_{ij}	= freight cost to ship product program $i \in \mathcal{C}^p$ from facility $j \in \mathcal{S}$ to assembly destination. This is the net present value cost over the life of the project, calculated as the annual freight cost multiplied by the NPV factor.
TC_{ij}	= tooling move cost to make product program $i \in \mathcal{C}^p$ in facility $j \in \mathcal{S}$.
MC_{pj}	= cost of moving any additional unit of capacity of process $p \in \mathcal{P}$ to facility $j \in \mathcal{S}$.
MPM_{pj}	= maximum number of units of capacity of process $p \in \mathcal{P}$ that can be moved to facility $j \in \mathcal{S}$.
CPU_{pj}	= capacity per unit of process $p \in \mathcal{P}$ at facility $j \in \mathcal{S}$.
RL_{ip}	= required load for requirement $i \in \mathcal{C}$ on process $p \in \mathcal{P}$.
SAC_{pj}	= available capacity of process $p \in \mathcal{P}$ at facility $j \in \mathcal{S}$.
$IPMC_{pj}$	= the total initial installed capacity for process $p \in \mathcal{P}$ at facility $j \in \mathcal{S}$. Note that $IPMC_{pj} - SAC_{pj}$ is the capacity already used by other parties at facility j .
IPP_j	= initial total number of processes having $IPMC_{pj} > 0$ available at facility $j \in \mathcal{S}^c$.

There were $\lambda = 42$ product programs, which we expanded into $n = 229$ manufacturing operations (also referred to as requirements) in order to improve granularity of manufacturing process assignments. Each product program could have multiple requirements, such as floor space and machine time requirements. In addition, for machine requirements, there were several machines on which a manufacturing requirement could be made, and the manufacturing requirements could be assigned to be made on several machines in the same facility. For example, an instrument panel could

be made on a 1,500, 2,000, or 2,500-ton press or any combination of the aforementioned.

3.2. Variables. For this MINLP, we have both continuous and binary decision variables. The first variable, x_{ij} , is a binary variable that determines if a requirement $i \in \mathcal{C}$ is assigned to facility $j \in \mathcal{S}$. In other words,

$$x_{ij} = \begin{cases} 1 & \text{if requirement } i \text{ is assigned to facility } j \\ 0 & \text{otherwise.} \end{cases}$$

The second variable, v_{ipj} , is the fraction of requirement $i \in \mathcal{C}$ produced on process $p \in \mathcal{P}$ at facility $j \in \mathcal{S}$. This variable enables partial loading of the total requirement across multiple machine sizes. For example, the required process hours could be split between 2000 ton and 2500 ton machines.

The third variable is a binary variable that specifies whether or not a requirement that is produced at a certain facility avoids double counting freight:

$$w_{ij} = \begin{cases} 1 & \text{if another product program that belongs to the same} \\ & \text{freight family as product program } i \in \mathcal{C}^p \text{ is produced at} \\ & \text{the same facility } j \in \mathcal{S} \\ 0 & \text{otherwise.} \end{cases}$$

Note that only one product program of any given product program group can get a freight discount. So, if product programs i_1 and i_2 are in the same freight family and both product program i_1 and i_2 are produced at facility j , then $w_{i_1j} = 1$ and $w_{i_2j} = 0 \forall j \in \mathcal{S}$ and $\forall i_1, i_2 \in \mathcal{C}^p$. For example, an instrument panel could receive reduced transportation costs by assigning cockpit assembly work to the same location as the base instrument panel and shipping a single finished assembly to the final assembly plant, as opposed to separately shipping the cockpit and instrument panel.

The fourth variable, m_{pj} , specifies how many units of capacity of type $p \in \mathcal{P}$ to move to supplier $j \in \mathcal{S}$. This is an integer variable that is greater than or equal to zero. Note that this variable is integer-constrained (see constraint (3.18)) by MPM_{pj} , which is introduced in Section 3.1. An example unit of capacity that could be moved to a facility is a 2,500 ton press.

The variable u_{pj} is a binary variable that keeps track of whether or not any additional units of capacity of process p were added to a facility j that did not already have any initial installed capacity of process type p .

$$u_{pj} = \begin{cases} 1 & \text{if any units of process } p \in \mathcal{P} \text{ are added to facility } j \in \mathcal{S}, \\ & \text{such that } IPMC_{pj} = 0 \\ 0 & \text{otherwise.} \end{cases}$$

The variable npp_j is the total number of processes available at facility $j \in \mathcal{S}$, including the “just moved in” processes: see constraint (3.13). It is a function of the initial total number of processes plus any units of new processes that are moved into a facility. Hence, we can let it be a continuous variable because it is the sum of integers.

The continuous variable pmc_{pj} is the maximum capacity for process $p \in \mathcal{P}$ at facility $j \in \mathcal{S}$, including any capacity that has been moved into facility j : see constraint (3.14).

The continuous variable ipu_j is the initial percent utilization for facility $j \in \mathcal{S}$: see constraint (3.18).

The continuous variable uc_{ij} is the unit cost for product program $i \in \mathcal{C}^p$ at facility $j \in \mathcal{S}$, which can be approximated by the function in constraint (3.16).

The continuous variable pc_{ij} is the production cost of a product program $i \in \mathcal{C}^p$ in facility $j \in \mathcal{S}$: see constraint (3.17).

For this MINLP, we had $\lambda = 42$ product programs, $n = 229$ requirements, $m = 57$ facilities and $r = 26$ processes. The MINLP has $m(n + \lambda + r) = 16,929$ binary variables, $rm = 1,482$ integer (non-binary) variables, and $m(nr + r + 2 + 2\lambda) = 345,762$ continuous variables, for a total of $m(n + nr + 3\lambda + 3r + 2) = 364,173$ variables. Note that the number of variables is an upper bound, as all facilities were not necessarily considered in all scenarios, and constraints (3.3) and (3.11) also reduce the number of variables.

3.3. Objective function. Our objective is to minimize all costs associated with the interiors business: product program producing, transporting, tooling, equipment moving, and facility closing. Including all of these costs allows the interiors business to capture not only traditional transportation and production costs, but also allows sourcing to occur at facilities to which equipment can be moved. The resulting nonlinear objective function is

$$\begin{aligned} \min_{\substack{w_{ij}, pc_{ij}, uc_{ij} \\ x_{ij}, v_{ipj}, m_{pj}, u_{pj} \\ \forall i \in \mathcal{C}, j \in \mathcal{S}, p \in \mathcal{P}}} & \sum_{\substack{i \in \mathcal{C}^p \\ j \in \mathcal{S}}} (FC_{ij} + TC_{ij})x_{ij} + \sum_{\substack{i \in \mathcal{C}^p \\ j \in \mathcal{S}}} pc_{ij}x_{ij} \\ - & \sum_{\substack{i \in \mathcal{C}^p \setminus \{i: CPM_i=0\}, \\ j \in \mathcal{S}}} FC_{ij}w_{ij} + \sum_{\substack{p \in \mathcal{P} \\ j \in \mathcal{S}^c}} MC_{pj}m_{pj} + \sum_{j \in \mathcal{S}^c} CC_j \end{aligned} \quad (3.1)$$

We describe the individual cost components in more details below:

- Freight costs, FC_{ij} , are incurred by shipping a product program $i \in \mathcal{C}^p$ from facility $j \in \mathcal{S}$. Since this cost is associated with a product program, it is carried by only one of the requirements; multiple entries for a product program representing additional requirements do not have additional costs. This is represented by $\sum_{i \in \mathcal{C}^p, j \in \mathcal{S}} FC_{ij}x_{ij}$, where the freight cost is realized only if the

product program i is assigned to a facility j (i.e., $x_{ij} = 1$). In addition, freight cost can be avoided for building common parts in the same facility. This is because they have the same volume shipped together or separately, so we can subtract the freight cost for additional common parts that are assigned to the same facility. This is captured by $-\sum_{i \in \mathcal{C}^p \setminus \{i : CPM_i=0\}, j \in \mathcal{S}} FC_{ij} w_{ij}$, where the freight cost is subtracted for additional common parts i that are shipped from facility j (i.e., this is indicated when $w_{ij} = 1$).

- The tooling move cost, TC_{ij} , is the cost to move tooling to produce product program $i \in \mathcal{C}^p$ in facility $j \in \mathcal{S}$. This is realized by $\sum_{i \in \mathcal{C}^p, j \in \mathcal{S}} TC_{ij} x_{ij}$, where tooling move cost is added only if product program i is assigned to facility j (i.e., $x_{ij} = 1$). $TC_{ij} = 0$ when sourcing a product program to its current production facility; however, when moving production to a different facility (even to a facility having the capacity available to make a product program), there is generally some tooling cost involved.
- The capacity move cost, MC_{pj} , is the cost to move a unit of capacity of process $p \in \mathcal{P}$ to facility $j \in \mathcal{S}$. This is to account for cost associated with moving capacity to a facility to either increase existing capacity or introduce new capacity. It is represented by $\sum_{p \in \mathcal{P}, j \in \mathcal{S}} MC_{pj} m_{pj}$, where cost is added only if a unit of capacity of process p is added to facility j (i.e., $m_{pj} > 0$).
- Production cost of a product program, pc_{ij} , is a variable that is dependent on capacity utilization at facility j , introducing nonlinearity into the objective function. It is captured by $\sum_{i \in \mathcal{C}^p, j \in \mathcal{S}} pc_{ij} x_{ij}$, where production cost is only charged for a product program i if it is produced in facility j (i.e., $x_{ij} = 1$).
- The facility closure cost, CC_j , is the cost to close facility j . Note that this is an input value and is added to the total objective function value if either Ford-owned facility, Utica or Saline, is chosen for closure (i.e., nothing is sourced to the Ford-owned facility). Ford does not have the ability to close any outside supplier facility and likewise would not incur closing cost if an outside supplier's facility closed.

3.4. Constraints. The following constraints for the MINLP convey the different business requirements:

- Each requirement must be assigned to exactly one facility ($n = 229$ constraints).

$$\sum_{j \in \mathcal{S}} x_{ij} = 1 \quad \forall i \in \mathcal{C} \quad (3.2)$$

- If requirements are in the same family, then they must be assigned to same facility ($\frac{mn(n-1)}{2} = 1,488,042$ constraints).

$$\begin{aligned} x_{i_1j} = x_{i_2j} \quad & \forall i_1, i_2 \in \mathcal{C}, j \in \mathcal{S} : \\ & i_2 < i_1 \wedge PF_{i_1} \neq 0 \wedge PF_{i_2} = PF_{i_1}. \end{aligned} \quad (3.3)$$

We guarantee that manufacturing requirements belonging to the same product program are sourced to the same facility by grouping them in the same “product-program family” and constraining those in the same product-program family to be sourced to the same facility.

- If a requirement i is produced at facility j , then the total fractional amount produced at j over all processes p should equal 1. In addition, if a requirement i is not produced at facility j , then no fractional amount of requirement i must be produced on a process p belonging to that facility j ($nm = 13,053$ constraints).

$$\sum_{p \in \mathcal{P}: RL_{ip} > 0} v_{ipj} = x_{ij} \quad \forall i \in \mathcal{C}, j \in \mathcal{S}. \quad (3.4)$$

- The capacity on every process p at a facility j must not be violated ($mr = 1,482$ constraints).

$$\begin{aligned} \sum_{i \in \mathcal{C}: RL_{ip} > 0} RL_{ip} v_{ipj} & \leq COF(SAC_{pj} + CPU_{pj} m_{pj}) \\ & \forall j \in \mathcal{S}, p \in \mathcal{P}. \end{aligned} \quad (3.5)$$

- Every requirement i must be fully assigned to one or more processes p at facility j ($n = 229$ constraints). Note that this constraint only needs to be for every requirement because constraints (3.2) and (3.4) guarantee that the total fractional amount of requirement i is produced only at one facility j . Although this constraint is implied by (3.2) and (3.4), it is useful for better defining the convex hull when solving the MIPs in Section 5.

$$\sum_{p \in \mathcal{P}: RL_{ip} > 0, j \in \mathcal{S}} v_{ipj} = 1 \quad \forall i \in \mathcal{C}. \quad (3.6)$$

- At least some threshold of business must be dedicated UAW facilities.

$$\sum_{i \in \mathcal{C}, j \in \mathcal{S}: U_j = 1} EH_i x_{ij} \geq UMIN. \quad (3.7)$$

- If product programs are in the same intellectual family, then they must be assigned to same holding company; that is, the facilities to which the product programs are assigned must belong to the same supplier ($\frac{\lambda(\lambda-1)}{2} = 861$ constraints).

$$\begin{aligned} \sum_{j \in \mathcal{S}} SU_j x_{i_1j} = \sum_{j \in \mathcal{S}} SU_j x_{i_2j} \quad & \forall i_1 \in \mathcal{C}^p : IF_{i_1} > 0, i_2 \in \mathcal{C}^p : \\ & IF_{i_1} = IF_{i_2} \wedge i_2 > i_1. \end{aligned} \quad (3.8)$$

As long as this constraint holds for $i_1, i_2 \in \mathcal{C}^p$, it will also hold for all other $i_1, i_2 \in \mathcal{C}$ because of constraint (3.3).

- If a product program i is not assigned to facility j , then the product program is not eligible for a freight discount to facility j ($\lambda m = 2,394$ constraints).

$$w_{ij} \leq x_{ij} \quad \forall i \in \mathcal{C}^p, j \in \mathcal{S} \quad (3.9)$$

- The following constraint forces the “produced together” variable, w_{ij} , to be zero if two product programs that get a benefit for being produced together are not produced at the same facility.

$$\begin{aligned} x_{i_1j} + x_{i_2j} &\geq 2w_{ij} \quad \forall j \in \mathcal{S}, i_1 \in \mathcal{C}^p : CPM_{i_1} > 0, i_2 \in \mathcal{C}^p : \\ &CPM_{i_1} = CPM_{i_2} \wedge i_2 > i_1. \end{aligned} \quad (3.10)$$

Similar to the previous constraint, we only consider $i_1, i_2 \in \mathcal{C}^p$ because a reduction in freight cost is only applicable to the unique product programs ($\frac{m\lambda(\lambda-1)}{2} = 49,077$ constraints).

- The following constraint guarantees that “produced together” variable, w_{ij} , is zero if a product program i_1 does not get a benefit for being produced together with other product programs at the same facility. It also forces w_{ij} to be zero if product program i_1 has a CPM_{i_1} value greater than zero that is the same as some other product program i_2 and $i_1 > i_2$. This constraint is useful in the presolve phase in eliminating variables ($\frac{m\lambda(\lambda-1)}{2} = 49,077$ constraints).

$$\begin{aligned} w_{ij} = 0 \quad \forall i_1, i_2 \in \mathcal{C}^p, j \in \mathcal{S} : (CPM_{i_1} = 0) \vee \\ (CPM_{i_1} > 0 \wedge CPM_{i_1} = CPM_{i_2} \wedge i_2 < i_1). \end{aligned} \quad (3.11)$$

- The following ensures that if requirement i is assigned to facility j , then the sum of all processes that requirement i uses in facility j satisfies requirement i 's demand ($nm = 13,053$ constraints):

$$\sum_{p \in \mathcal{P}: RL_{ip} > 0} RL_{ip} v_{ipj} = PR_i x_{ij} \quad \forall i \in \mathcal{C}, j \in \mathcal{S}. \quad (3.12)$$

- The following keeps track of the number of unique process types available at a facility. It adds the number of new processes moved to a facility (does not include adding more process units to existing processes) to the initial number of processes at a facility ($m = 57$ constraints).

$$npp_j = IPP_j + \sum_{p \in \mathcal{P}} u_{pj} \quad \forall j \in \mathcal{S}. \quad (3.13)$$

- The constraint below updates the maximum capacity for a process at a facility, updating the initial total capacity with the total capacity that is moved into the facility ($rm = 1,482$ constraints).

$$pmc_{pj} = IPMC_{pj} + CPU_{pj}m_{pj} \quad \forall p \in \mathcal{P}, j \in \mathcal{S}. \quad (3.14)$$

- The following nonlinear constraint keeps track of the initial percent utilization of a facility, accounting for any new capacity that was added to the facility ($m = 57$ constraints).

$$ipu_j = \left(\sum_{p \in \mathcal{P}} \frac{pmc_{pj} - SAC_{pj}}{pmc_{pj}} \right) / npp_j \quad \forall j \in \mathcal{S}. \quad (3.15)$$

- The following nonlinear constraint approximates the unit cost, where A_{ij} is the unit variable cost at this facility, and B_{ij} is the facility fixed cost attributed to a unit at 100% utilization of all processes at the facility ($\lambda m = 2,394$ constraints). The facility congestion costs are not accounted for in this approximation.

$$uc_{ij} = A_{ij} + \frac{B_{ij}}{\sum_{p \in \mathcal{P}, i \in \mathcal{C}} v_{ipj} + ipu_j} \quad \forall i \in \mathcal{C}^p, j \in \mathcal{S}. \quad (3.16)$$

- The following constraint defines the production cost of a product program assigned to facility ($\lambda m = 2,394$ constraints):

$$pc_{ij} = NPV \cdot uc_{ij} \frac{VV_i}{1,000} \quad \forall i \in \mathcal{C}^p, j \in \mathcal{S}. \quad (3.17)$$

Note that we divide VV_i by a thousand to convert to the same units of measure: pc_{ij} is in millions of dollars, uc_{ij} is in dollars, and VV_i is in thousand of units per year.

- The following constraint guarantees that $u_{pj} = 1$ if any units of capacity of a process are moved into a facility when there is no initial installed capacity for that process.

$$m_{pj} \leq MPM_{pj}u_{pj} \quad \forall p \in \mathcal{P}, j \in \mathcal{S} : IPMC_{pj} = 0. \quad (3.18)$$

If there is initial installed capacity for a process, then $u_{pj} = 0$ (together with constraint 3.18, it has $rm = 1,482$ constraints).

$$u_{pj} = 0 \quad \forall p \in \mathcal{P}, j \in \mathcal{S} : IPMC_{pj} > 0. \quad (3.19)$$

- The following constraint guarantees that if no units of process are moved to a facility, then there are no units of process p that were added to facility j ($rm = 1,482$ constraints).

$$u_{pj} \leq m_{pj} \quad \forall p \in \mathcal{P}, j \in \mathcal{S}. \quad (3.20)$$

This MINLP has a nonlinear objective function and several nonlinear constraints, (3.15)-(3.17). Altogether, there are $\frac{mn(n-1)}{2} + \frac{\lambda(\lambda-1)(2m+1)}{2} + m(3\lambda + 2n + 4r + 2) + 2n + 1 = 1,626,846$ constraints.

4. Discrete MINLP reformulation. The MINLP problem, as described in Section 3, has a nonlinear objective function, several hundred thousand variables, and over a million constraints (a subset of which is nonlinear): this MINLP is too large to solve with current MINLP solution techniques. With a very tight deadline in which we had to solve this problem and deliver an analysis (two months), we focused on developing a formulation that could be easily solved using existing solution techniques. Our goal was to remove the nonlinearity in both the objective function and constraints so that we could reduce the MINLP to a MIP, allowing the use of available solvers capable of tackling problems with millions of variables and constraints. In this section, we describe how we first reduced the complexity of the problem by removing the nonlinearity in the objective function.

Recall that the nonlinearity in the objective function was caused by how we derived the non-standard unit cost of a product program. In this section, we describe how we linearized this cost by approximating the facilities' utilization curves into discrete steps having constant cost factors. To get the product program's utilization-adjusted unit cost, we multiply those factors by the variable unit cost. Because the nonlinear cost curves, themselves, are estimates, the approach of using the utilization levels adequately represented the business problem. To validate this assumption, we ran a calibration scenario using the existing sourcing pattern at ACH facilities to confirm agreement with current budget revenues: tests found that we were within five percent accuracy.

We refer to the resulting model as the “reformulated MINLP” because although we remove the nonlinearity in the objective function, there are still nonlinear constraints. We will first describe the new and modified inputs required for this reformulated MINLP. Next, we describe the additional dimension to some of the variables and the new variables for this MINLP that reflect the utilization range choice. We conclude with a discussion of the added and modified constraints.

4.1. Input Information. The list below describes the *additional* inputs that were necessary for the MINLP reformulation.

\mathcal{Q}	= $\{1, \dots, \varrho\}$ set of utilization ranges for facilities.
QC_{qj}	= production cost multiplier for facility $j \in \mathcal{S}$ when the facility is in utilization range $q \in \mathcal{Q}$.
UB_q	= upper bound on utilization range $q \in \mathcal{Q}$; $UB_0 = 0$.
UBC_i	= base unit cost of product program $i \in \mathcal{C}^p$.
UC_i	= calculated unit cost of product program $i \in \mathcal{C}^p$.
PC_{ij}	= calculated production cost of product program $i \in \mathcal{C}^p$ at

Note that UC_i and PC_{ij} were previously the variables uc_i and pc_{ij} in Section 3 but are now parameters in the reformulated MINLP.

4.2. Variables. Many of the variables for the reformulated MINLP are the same as the variables in Section 3.2; exceptions include the variables x_{iqj} and v_{ipqj} that now have the dimension q added to designate the utilization range for a facility. We also introduce a new variable y_{qj} that will determine which utilization range is chosen for a facility.

So, the variable x_{ij} is now redefined as

$$x_{iqj} = \begin{cases} 1 & \text{if requirement } i \in \mathcal{C} \text{ is produced in utilization range } q \in \mathcal{Q} \\ & \text{by facility } j \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases}$$

The variable v_{ipj} is now redefined as v_{ipqj} , which allows a fraction of requirement $i \in \mathcal{C}$ to be produced on different processes $p \in \mathcal{P}$ at facility $j \in \mathcal{S}$ in utilization range $q \in \mathcal{Q}$.

The new variable, y_{qj} , is a binary variable that determines whether or not a facility's total production is within a certain utilization range q . The utilization level is only defined for the facilities to which we source a share of our own business; we are ambivalent regarding the utilization of a facility to which we have not sourced any business. We write this as,

$$y_{qj} = \begin{cases} 1 & \text{if facility } j \in \mathcal{S} \text{ to which we source a share of our own} \\ & \text{business has utilization levels in range } q \in \mathcal{Q}, \\ 0 & \text{otherwise.} \end{cases}$$

Assuming values for m , n , λ , and r have the same values as in Section 3.2, and $\varrho = 10$, there are $m(n\varrho + \lambda + r + \varrho) = 134,976$ binary variables, $rm = 1,482$ integer (non-binary) variables, and $m(nr\varrho + r + 2 + 2\lambda) = 3,400,164$ continuous variables, with a total of $m(n\varrho + nr\varrho + 3\lambda + 3r + 2 + \varrho) = 3,536,622$ variables. Note that this is an upper bound on the number of variables, as not all facilities are necessarily considered in each scenario, and constraints (3.3) and (3.11) also reduce the number of variables.

4.3. Objective function. The objective function for the reformulated MINLP is similar to the one in Section 3.3. However, the production cost for a product program is now a fixed input that depends on the facility's utilization range; this makes the objective function linear. This is because the unit cost variable, uc_{ij} , from the reformulated MINLP has been approximated by $UC_{iqj} = UBC_i \cdot QC_{qjj}$ for some $q_j \in \mathcal{Q}$ and for all $j \in \mathcal{S}$ such that $y_{qjj} = 1$: see Figure 1. Hence, the production cost of a product program, PC_{iqj} , is $(NPV \cdot VVi)UC_{iqj}/1000$.

4.4. Constraints. As a result of linearizing the objective function, we now have some modified and additional constraints in the reformulated MINLP. Constraints (3.2) through (3.10) and (3.12) have been modified from the constraints discussed in Section 3.4 to include the new dimension associated with the utilization range q . For every variable that contains a q subscript, there is a summation over q in the constraint for that variable, with the exception of constraint (3.4) that is for every q .

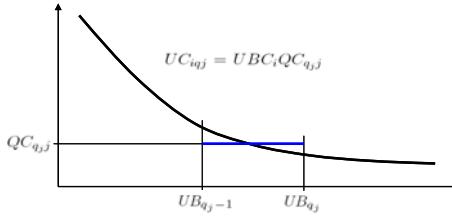


FIG. 1. We discretized each facility's utilization curve into ranges; range length could be unequal to better approximate the curve. In this diagram, the utilization range, q_j , is defined by the interval from UB_{qj-1} to UB_{qj} and has a unit cost factor $QC_{qj} \forall j \in \mathcal{S}$.

The following constraints are new constraints that were not in the original MINLP described in Section 3.4. They have been added as a result of introducing the utilization range approximation to the facility cost curve.

- A facility j can use at most one utilization level q . If a facility is not chosen for Ford's business (even though the facility may carry third-party business), then we specify that no utilization range at that facility is chosen ($m = 57$ constraints).

$$\sum_q y_{qj} \leq 1 \quad \forall j \in \mathcal{S} \quad (4.1)$$

- Facility utilization, defined as an average of all installed process utilizations, has a known upper bound in each utilization range q , which we enforce for each facility with the following nonlinear constraint ($m\varrho = 570$ constraints).

$$\begin{aligned} & \sum_{i \in \mathcal{C}, p \in \mathcal{P}: pmc_{p,j} > 0 \wedge RL_{ip} > 0} \frac{RL_{ip}}{npp_j pmc_{pj}} v_{ipqj} \\ & \leq (UB_q - ipu_j) y_{qj} \quad \forall j \in \mathcal{S}, q \in \mathcal{Q} \end{aligned} \quad (4.2)$$

- Similarly, each utilization range has a known lower bound utilization, which we enforce for each facility with the following nonlinear constraint ($(m\varrho - 1) = 513$ constraints).

$$\begin{aligned} & \sum_{i \in \mathcal{C}, p \in \mathcal{P}: pmc_{p,j} > 0 \wedge RL_{ip} > 0} \frac{RL_{ip}}{npp_j pmc_{pj}} v_{ipqj} \\ & \geq (UB_{q-1} - ipu_j) y_{qj} \quad \forall j \in \mathcal{S}, q \in \mathcal{Q} : q > 1 \end{aligned} \quad (4.3)$$

- The next two constraints are linking constraints. The first guarantees that if no product program i is sourced to a facility j in

utilization range q , then utilization range q for facility j can not be selected ($m\varrho = 570$ constraints).

$$y_{qj} \leq \sum_{i \in \mathcal{C}} x_{iqj} \quad \forall j \in \mathcal{S}, q \in \mathcal{Q}. \quad (4.4)$$

The second guarantees that if at least one product program i is sourced to a facility j in utilization range q , then utilization range q for facility j must be selected. ($nm\varrho = 130,530$ constraints).

$$x_{iqj} \leq y_{qj} \quad \forall i \in \mathcal{C}, q \in \mathcal{Q}, j \in \mathcal{S}. \quad (4.5)$$

- If a product program i is assigned to a process p at facility j then it also must be assigned to a utilization range q at plant j ($nm = 13,053$ constraints).

$$\sum_{p \in \mathcal{P}: RL_{ip} > 0, q \in \mathcal{Q}} RL_{ip} v_{ipqj} = PR_i \sum_{q \in \mathcal{Q}} x_{iqj} \quad \forall i \in \mathcal{C}, j \in \mathcal{S}. \quad (4.6)$$

This reformulated MINLP has a linear objective function and several non-linear constraints, (3.15) through (3.17) from the original MINLP plus (4.2) and (4.3). In total, there are $\frac{mn(n-1)}{2} + \frac{\lambda(\lambda-1)(2m+1)}{2} + m(3\lambda + 2n + 4r + 2n\varrho + 3\varrho + 2) + 2n + 1 = 1,889,616$ constraints in this discrete MINLP reformulation. Recall that this is an upper bound on the number of constraints since not all facilities are considered in each scenario.

5. Solution technique. Although we were able to remove the nonlinearity in the objective function to create the reformulated MINLP described in the previous section, we are still left with a large-scale MINLP where the discrete and continuous variables are non-separable. In this section, we describe how we obviate the nonlinearity in the constraints of the reformulated MINLP by iteratively fixing different variables in the reformulated MINLP, allowing us to solve a MIP at each iteration. By carefully selecting which variable to fix at each iteration, we not only are able to solve MIPs at each iteration, but we are also able to reduce the number of constraints and variables in each formulation. We first introduce the two MIPs that we iteratively solve, each of which captures a subset of the decisions to be made. Next, we describe the algorithm that provides us with a solution to the discrete MINLP. Finally, we discuss convergence properties of the algorithm.

5.1. Decoupling the MINLP. We introduce a decoupling of the discrete MINLP model into two MIPs that can then be solved iteratively until convergence to a solution of the discrete MINLP reformulation (see [Figure 2](#)). Each MIP is actually a subset of the reformulated MINLP with a different subset of variables fixed in the different MIPs. We chose this approach because solution techniques for MIPs are well defined for

efficiently solving problems with millions of variables and constraints [6]. Not only do we avoid the nonlinearity in the constraints, but we are also able to reduce the number of variables and constraints needed to describe the problem.

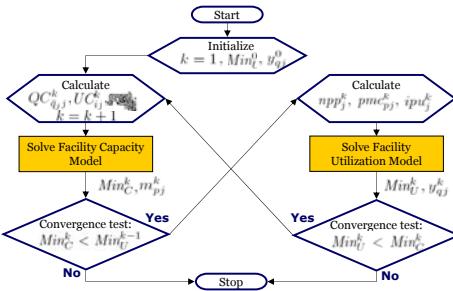


FIG. 2. Flowchart that describes the solution technique.

The first MIP model we refer to as the Facility Capacity Model (FCM); it assumes that the utilization range for a supplier is fixed, and it solves to determine to which facilities and processes to source and whether or not facilities that do not currently have capacity should have capacity added. This solution will then determine upper bounds on a facility's process capacity. The second model we refer to as the Facility Utilization Model (FUM). This model accepts as inputs a facility's upper bound on capacity for each process, the number of unique processes at each facility, and the initial facility percent utilization and determines which utilization range a facility falls within, as well as the same sourcing decisions. We conclude this section with a detailed discussion of the algorithm in Figure 2.

5.1.1. Facility capacity problem formulation. The FCM MIP assumes that the utilization range for each supplier is fixed; hence, we remove the subscript q from all variables. The FCM answers the following two questions: to which facilities and processes to source, and whether or not facilities that do not currently have capacity should have capacity added. As stated in Section 2, most categories of ACH-owned equipment could be moved from Saline or Utica to a new supplier's site, at a certain expense. This FCM solution will then determine upper bounds on a facility's process capacity. The FCM model differs from the reformulated MINLP in that it does not include the variable y_{qj} and so does not need the associated bounds or the constraints associated with this variable in (4.1)–(4.6). Additionally, constraints (3.13) through (3.15) are not needed because npp_j , pmc_{pj} , and ipu_j are calculated after the FCM is solved. Constraints (3.16) and (3.17) are not needed because PC_{iqj} and UC_{iqj} are not decision variables and are set according to the fixed utilization range from the previous solve of the FUM. Constraints (3.18) through (3.20) are no longer needed

because u_{pj} is determined by testing to see if $m_{pj}^k > 0$ and $IPMC_{pj} = 0$ after the FCM is solved.

The formulation below defines the objective function solution Min_C^k for the FCM at iteration k .

This MIP has $m(n+nr+\lambda+r) = 356,307$ variables and $\frac{\lambda(\lambda-1)(m+2)}{2} + \frac{mn(n-1)}{2} + 2n(m+1) + mr + 1 = 1,566,888$ constraints before the pre-solve. The number of variables and constraints that can be eliminated during CPLEX's pre-solve is dependent upon the scenario (e.g. number of external facilities we consider, number of closed for business facilities, etc.); in one scenario, the MIP had 5,864 variables and 2,575 constraints after the pre-solve.

$$\begin{aligned} Min_C^k = & \min_{\substack{w_{ij} \\ x_{ij}, \\ v_{ipj}, m_{pj} \\ \forall i \in \mathcal{C}, j \in \mathcal{S}, p \in \mathcal{P}}} \sum_{i \in \mathcal{C}, j \in \mathcal{S}} (FC_{ij} + TC_{ij} + PC_{ij}^{k-1})x_{ij} \\ & - \sum_{\substack{i \in \mathcal{C} \setminus \{i: CPM_i=0\}, \\ j \in \mathcal{S}}} FC_{ij}w_{ij} + \sum_{p \in \mathcal{P}, j \in \mathcal{S}} MC_{pj}m_{pj} + \sum_{j \in \mathcal{S}^c} CC_j \end{aligned}$$

s.t.

$$\sum_{j \in \mathcal{S}} x_{ij} = 1 \quad \forall i \in \mathcal{C} \quad (5.1)$$

$$\begin{aligned} x_{i_1j} &= x_{i_2j} \quad \forall i_1, i_2 \in \mathcal{C}, j \in \mathcal{S}: \\ i_2 < i_1 \wedge PF_{i_1} &\neq 0 \wedge PF_{i_2} = PF_{i_1} \end{aligned} \quad (5.2)$$

$$\sum_{p \in \mathcal{P}: RL_{ip} > 0} v_{ipj} = x_{ij} \quad \forall i \in \mathcal{C}, j \in \mathcal{S} \quad (5.3)$$

$$\begin{aligned} \sum_{i \in \mathcal{C}: RL_{ip} > 0} RL_{ip}v_{ipj} &\leq COF(SAC_{pj} + CPU_{pj}m_{pj}) \\ \forall j \in \mathcal{S}, p \in \mathcal{P} \end{aligned} \quad (5.4)$$

$$\sum_{p \in \mathcal{P}: RL_{ip} > 0, j \in \mathcal{S}} v_{ipj} = 1 \quad \forall i \in \mathcal{C} \quad (5.5)$$

$$\sum_{i \in \mathcal{C}, j \in \mathcal{S}: U_j = 1} EH_i x_{ij} \geq UMIN \quad (5.6)$$

$$\sum_{j \in \mathcal{S}} SU_j x_{i_1 j} = \sum_{j \in \mathcal{S}} SU_j x_{i_2 j} \quad \forall i_1 \in \mathcal{C}^p : IF_{i_1} > 0, \\ i_2 \in \mathcal{C}^p : IF_{i_1} = IF_{i_2} \wedge i_2 > i_1 \quad (5.7)$$

$$w_{ij} \leq x_{ij} \quad \forall i \in \mathcal{C}^p, j \in \mathcal{S} \quad (5.8)$$

$$x_{i_1 j} + x_{i_2 j} \geq 2w_{i_1 j} \quad \forall j \in \mathcal{S}, i_1 \in \mathcal{C}^p : CPM_{i_1} > 0, \\ i_2 \in \mathcal{C}^p : CPM_{i_1} = CPM_{i_2} \wedge i_2 > i_1 \quad (5.9)$$

$$w_{i_1 j} = 0 \quad \forall i_1, i_2 \in \mathcal{C}^p, j \in \mathcal{S} : (CPM_{i_1} = 0) \vee \\ (CPM_{i_1} > 0 \wedge CPM_{i_1} = CPM_{i_2} \wedge i_2 < i_1) \quad (5.10)$$

$$\sum_{p \in \mathcal{P}: RL_{ip} > 0} RL_{ip} v_{ipj} = PR_i x_{ij} \quad \forall i \in \mathcal{C}, j \in \mathcal{S} \quad (5.11)$$

$$x_{ij} \in \mathcal{B} \quad \forall i \in \mathcal{C}, j \in \mathcal{S} \quad (5.12)$$

$$0 \leq v_{ipj} \leq 1 \quad \forall i \in \mathcal{C}, p \in \mathcal{P}, j \in \mathcal{S} \quad (5.13)$$

$$w_{ij} \in \mathcal{B} \quad \forall i \in \mathcal{C}^p, j \in \mathcal{S} \quad (5.14)$$

$$0 \leq m_{pj} \leq MPM_{pj} \quad \forall p \in \mathcal{P}, j \in \mathcal{S}. \quad (5.15)$$

5.1.2. Facility utilization problem formulation. The FUM accepts as an input each facility's upper bound on capacity for each process (determined by the variable m_{pj}^k in the FCM) and decides which utilization range a facility falls within, as well as the same sourcing decisions. As mentioned in Section 2, a manufacturing site's capacity utilization impacts its cost structure; this model allows a supplier that runs a plant at higher capacity to incur a lower per-unit cost than an under-utilized facility due to improved fixed cost coverage.

The formulation below defines the objective function solution Min_U^k for the FUM at iteration k . The FUM differs from the reformulated MINLP since the variables m_{pj}^k , npp_j^k , pmc_{pj}^k , u_{pj}^k , and ipu_j^k from the reformulated MINLP are constants. The value for m_{pj}^k is passed from the FCM and is used to determine u_{pj}^k , npp_j^k , pmc_{pj}^k , and ipu_j^k . Hence, constraints (4.2) and (4.3), which are (5.28) and (5.29) in the FUM, respectively, are now linear. This MIP has $m\varrho(nr + n + 1) + \lambda m = 3,527,274$ variables and $\frac{mn(n-1)}{2} + \frac{\lambda(\lambda-1)(2m+1)}{2} + n + m(3\varrho + \lambda + 2n + r + 1) + 1 = 1,619,036$ constraints before the pre-solve; The number of variables and constraints that can be eliminated during CPLEX's pre-solve is dependent upon the scenario; in one scenario, the MIP had 14,741 variables and 4,940 constraints after the pre-solve.

$$\begin{aligned}
Min_U^k = & \min_{\substack{x_{iqj}, y_{qj}, v_{ipqj}, \\ \forall i \in \mathcal{C}, j \in \mathcal{S}, q \in \mathcal{Q}, p \in \mathcal{P}; \\ w_{ij} \forall i \in \mathcal{C}^p, j \in \mathcal{S}}} \sum_{i \in \mathcal{C}, j \in \mathcal{S}, q \in \mathcal{Q}} (FC_{ij} + TC_{ij} + PC_{iqj}) x_{iqj} \\
& - \sum_{\substack{i \in \mathcal{C} \setminus \{i : CPM_i = 0\}, \\ j \in \mathcal{S}}} FC_{ij} w_{ij} + \sum_{p \in \mathcal{P}, j \in \mathcal{S}} MC_{pj} m_{pj}^k + \sum_{j \in \mathcal{S}^c} CC_j
\end{aligned}$$

s.t.

$$\sum_{q \in \mathcal{Q}, j \in \mathcal{S}} x_{iqj} = 1 \quad \forall i \in \mathcal{C} \quad (5.16)$$

$$\begin{aligned}
\sum_{q \in \mathcal{Q}} x_{i_1 q j} &= \sum_{q \in \mathcal{Q}} x_{i_2 q j} \quad \forall i_1, i_2 \in \mathcal{C}, \quad j \in \mathcal{S}: \quad i_2 < i_1 \wedge PF_{i_1} \neq 0 \\
&\quad \wedge PF_{i_2} = PF_{i_1} \quad (5.17)
\end{aligned}$$

$$\sum_{p \in \mathcal{P}: RL_{ip} > 0} v_{ipqj} = x_{iqj} \quad \forall i \in \mathcal{C}, \quad j \in \mathcal{S}, \quad q \in \mathcal{Q} \quad (5.18)$$

$$\begin{aligned}
\sum_{i \in \mathcal{C}: RL_{ip} > 0, q \in \mathcal{Q}} RL_{ip} v_{ipqj} &\leq COF(SAC_{pj} + CPU_{pj} m_{pj}^k) \sum_{q \in \mathcal{Q}} y_{qj} \\
&\quad \forall j \in \mathcal{S}, p \in \mathcal{P} \quad (5.19)
\end{aligned}$$

$$\sum_{p \in \mathcal{P}: RL_{ip} > 0, q \in \mathcal{Q}, j \in \mathcal{S}: SAC_{pj} > 0} v_{ipqj} = 1 \quad \forall i \in \mathcal{C} \quad (5.20)$$

$$\sum_{i \in \mathcal{C}, q \in \mathcal{Q}, j \in \mathcal{S}: U_j = 1} EH_i x_{iqj} \geq UMIN \quad (5.21)$$

$$\begin{aligned}
\sum_{q \in \mathcal{Q}, j \in \mathcal{S}} SU_j x_{i_1 q j_1} &= \sum_{q \in \mathcal{Q}, j \in \mathcal{S}} SU_j x_{i_2 q j_2} \quad \forall i_1 \in \mathcal{C}^p: IF_{i_1} > 0, \\
&\quad i_2 \in \mathcal{C}^p: IF_{i_1} = IF_{i_2} \wedge i_2 > i_1 \quad (5.22)
\end{aligned}$$

$$w_{ij} \leq \sum_{q \in \mathcal{Q}} x_{iqj} \quad \forall i \in \mathcal{C}^p, \quad j \in \mathcal{S} \quad (5.23)$$

$$\sum_{q \in \mathcal{Q}} x_{i_1 q j} + \sum_{q \in \mathcal{Q}} x_{i_2 q j} \geq 2w_{i_1 j} \quad \forall j \in \mathcal{S}, \quad i_1 \in \mathcal{C}^p : CPM_{i_1} > 0,$$

$$i_2 \in \mathcal{C}^p : CPM_{i_1} = CPM_{i_2} \wedge i_2 > i_1 \quad (5.24)$$

$$w_{i_1 j} = 0 \quad \forall i_1, i_2 \in \mathcal{C}^p, \quad j \in \mathcal{S} : (CPM_{i_1} = 0) \vee \\ (CPM_{i_1} > 0 \wedge CPM_{i_1} = CPM_{i_2} \wedge i_2 < i_1) \quad (5.25)$$

$$\sum_{p \in \mathcal{P} : RL_{ip} > 0, q \in \mathcal{Q}} RL_{ip} v_{ipqj} = PR_i \sum_{q \in \mathcal{Q}} x_{iqj} \quad \forall i \in \mathcal{C}, \quad j \in \mathcal{S} \quad (5.26)$$

$$\sum_q y_{qj} \leq 1 \quad \forall j \in \mathcal{S} \quad (5.27)$$

$$\sum_{i \in \mathcal{C}, p \in \mathcal{P} : pmc_{p,j} > 0 \wedge RL_{ip} > 0} \frac{RL_{ip}}{npp_j^k pmc_{pj}^k} v_{ipqj} \leq (UB_q - ipu_j^k) y_{qj} \\ \forall j \in \mathcal{S}, q \in \mathcal{Q} \quad (5.28)$$

$$\sum_{i \in \mathcal{C}, p \in \mathcal{P} : pmc_{p,j} > 0 \wedge RL_{ip} > 0} \frac{RL_{ip}}{npp_j^k pmc_{pj}^k} v_{ipqj} \geq \\ (UB_{q-1} - ipu_j^k) y_{qj} \quad \forall j \in \mathcal{S}, q \in \mathcal{Q} : q > 1 \quad (5.29)$$

$$y_{qj} \leq \sum_{i \in \mathcal{C}} x_{iqj} \quad \forall j \in \mathcal{S}, q \in \mathcal{Q} \quad (5.30)$$

$$x_{iqj} \leq y_{qj} \quad \forall i \in \mathcal{C}, q \in \mathcal{Q}, j \in \mathcal{S} \quad (5.31)$$

$$x_{iqj} \in \mathcal{B} \quad \forall i \in \mathcal{C}, \quad q \in \mathcal{Q}, j \in \mathcal{S} \quad (5.32)$$

$$y_{qj} \in \mathcal{B} \quad \forall q \in \mathcal{Q}, j \in \mathcal{S} \quad (5.33)$$

$$0 \leq v_{ipqj} \leq 1 \quad \forall i \in \mathcal{C}, \quad p \in \mathcal{P}, \quad q \in \mathcal{Q}, j \in \mathcal{S} \quad (5.34)$$

$$w_{ij} \in \mathcal{B} \quad \forall i \in \mathcal{C}^p, j \in \mathcal{S} \quad (5.35)$$

5.2. Algorithm. In this section, we describe in detail the algorithm we used to solve the reformulated MINLP. We will refer to the FUM and FCM models described above and the flowchart of the algorithm in [Figure 2](#) on page 611. This algorithm allows us to solve the reformulated MINLP by fixing certain variables in the reformulated MINLP at each stage in the iteration, resulting in solving only MIPs (i.e., the FCM and FUM).

5.2.1. Initialization. The first step in the algorithm in [Figure 2](#) is to initialize parameters. We set the iteration number $k = 1$, initialize the starting FUM objective function value, Min_U^0 , to an arbitrarily large value, and set the utilization level, y_{qj}^0 , for each facility $j \in \mathcal{S}$. We performed testing to determine the best initialization for y_{qj}^0 that would yield the fastest convergence. We tried the following three strategies: starting from the highest, the current, and the lowest facility utilization range. After much testing, the results demonstrated that we achieved the fastest convergence on average when using the highest facility utilization for y_{qj}^0 ; that is, we assumed that all facilities (no matter whether we sourced or never sourced any business to such a facility in the past) were in their last (i.e. highest) utilization range:

$$y_{qj}^0 = 0 \quad \forall q \in \mathcal{Q} : q \neq \varrho, \quad j \in \mathcal{S};$$

$$y_{\varrho,j}^0 = 1 \quad \forall j \in \mathcal{S}.$$

5.2.2. Calculate input parameters for FCM. The next step in the algorithm in [Figure 2](#) is to calculate the product program costs at all facilities based on the utilization level that was either assumed in Section 5.2.1 for each facility, or determined in Section 5.2.6 for facilities having a share of Ford business.

But first, note that if FUM determined that facility j does *not* have a share of Ford business, then the value $y_{qj}^k = 0 \forall q \in \mathcal{Q}$ will be passed from the FUM. We need to adjust these values to account for the non-Ford initial utilization:

```

For  $j = 1, \dots, m$ 
   $t = 0$ 
  For  $q = 1, \dots, \varrho$ 
     $t = t + y_{qj}^k$ 
  End For
  If ( $t = 0$ ) then
    For  $q = 1, \dots, \varrho$ 
      If ( $UB_{q-1} < ipu_j^k \leq UB_q$ ) then
         $y_{qj}^k = 1$ 
      End If
    End For
  End If
End For

```

Once this is done, we can determine the cost multiplier for each facility: if facility j is in utilization level q_j (that is, if $y_{q_j,j}^k = 1$), then we use cost multiplier $QC_{q_j,j}^k$ for that facility. The value of $QC_{q_j,j}^k$ is used consequently to determine the unit cost for each product program:

$$UC_{ij}^k = UBC_i \cdot QC_{q_j,j}^k \quad \forall i \in \mathcal{C}^p, j \in \mathcal{S};$$

$$PC_{ij}^k = NPV \cdot UC_{ij}^k \frac{VV_i}{1,000} \quad \forall i \in \mathcal{C}^p, j \in \mathcal{S}.$$

Recall that we approximate UC_{ij}^k as described in Section 4.3 and calculate PC_{ij}^k as defined in (3.17). Finally, we update the iteration counter (i.e., $k = k + 1$), and return to the step in Section 5.2.3.

5.2.3. Solving the FCM. The third step in the algorithm in Figure 2 is to solve the FCM, which will tell us how many units of process capacity is moved to a facility.

This MIP typically took around 10 minutes to solve. The output from this model is Min_C^k , m_{pj}^k , w_{ij} , x_{ij} , and v_{ipj} : note that we only pass Min_C^k and m_{pj}^k , the objective function value and the process capacity moves to each facility, respectively, to the next step in the algorithm. The value of m_{pj}^k will be used to determine the upper bound on process capacity at the k^{th} iteration in the FUM.

5.2.4. Convergence check after FCM. The fourth step in the algorithm in Figure 2 is to check for convergence. If $Min_C^k < Min_U^{k-1}$ (i.e., we obtained a smaller objective function value for the FCM than we did for the FUM in the previous iteration), then the algorithm continues. If not, then the algorithm has converged.

5.2.5. Calculate input parameters for FUM. The fifth step in the algorithm in Figure 2 is to calculate the input parameters necessary for the FUM that are functions of or dependent upon m_{pj}^k . First, we determine if any new processes were added to a facility for which there was no prior processes of that type. In the reformulated MINLP, the variable u_{pj} was a binary variable that established this. However, we can now determine if any new processes were added to a facility for which there was no prior processes of that type with the following check: if $IPMC_{pj} = 0 \wedge m_{pj}^k > 0$, then no prior process of that type previously existed, and now it has been added. Next, we need to calculate npp_j^k : constraint (3.13) in the reformulated MINLP was previously used to determine this. We can now update npp_j^k after solving the FCM by iterating over all plants j and all processes p , updating npp_j^k only if for any $m_{pj}^k > 0$, $IPMC_{pj} = 0$. Subsequently, we can update pmc_{pj} and ipu_j . That is, we update these parameters as follows:

Let $npp_j^k = IPP_j$

For $j = 1, \dots, m$

For $p = 1, \dots, r$

$$pmc_{pj} = IPMC_{pj} + CPU_{pj}m_{pj}^k$$

```

If  $IPMC_{pj} = 0 \wedge m_{pj}^k > 0$  then
     $npp_j^k = npp_j^k + 1$ 
End If
End For
 $ipu_j^k = \left( \sum_{p \in \mathcal{P}} \frac{pmc_{pj} - SAC_{pj}}{pmc_{pj}} \right) / npp_j$ 
End For

```

Recall that we calculate pmc_{pj}^k as defined in (3.14) and ipu_j^k as defined in (3.18).

5.2.6. Solving the FUM. The sixth step in the algorithm in Figure 2 is to solve the FUM. We pass the values Min_C^k , m_{pj}^k , npp_j^k , pmc_{pj}^k , and ipu_j^k to the FUM. These values provide an upper bound on the process capacity at each facility, the facility's initial percent utilization, and the number of processes at each facility. Initially, this MIP took between 20 and 30 minutes to solve using the dual simplex method. However, we changed to an interior point solver and reduced the solve time by half.

The output of this model is Min_U^k , w_{ij}^k , x_{igj}^k , y_{qj}^k , and v_{ipqj}^k . Note that we only pass Min_U^k and y_{qj}^k , the objective function value and the utilization range for each facility, respectively, to the next step in the algorithm.

5.2.7. Convergence check after FUM. After solving the FUM, the seventh step in the algorithm in Figure 2 is to check for convergence. If $Min_U^k < Min_C^k$ (i.e., we obtained a smaller objective function value for the FUM than we did for the FCM in this iteration), then the algorithm continues. If not, then the algorithm has converged.

5.2.8. Implementation. The two MIP models, Facility Capacity (section 5.1.1) and Facility Utilization (section 5.1.2) models were implemented in ILOG OPL Studio 5.2 [3], using CPLEX 10.2 with Microsoft Excel 2003 as the user interface. The tool was run on a 3.0 GHz Intel Xeon dual-processor workstation with 2GB RAM.

5.2.9. Algorithm convergence. We will now show that our algorithm is guaranteed to converge. We introduce the following notations for clarity. We define the vector z_C^k to be the solution to the FCM model at the k^{th} iteration the vector z_U^k be the solution to the FUM model at the k^{th} iteration. We let the optimal solution be F^* and update its value after solving the FCM or FUM.

CPLEX's MIP solver produces a global optimal solution for both the FUM and FCM at each iteration k ; [2] provides details on convergence theory for the barrier and dual simplex method. In addition, at the k^{th} iteration, the solution for FCM^k is also feasible for FUM^k because z_C^k satisfies the constraints of FUM^k for the fixed utilization range q . Because z_C^k is feasible for FUM^k , it follows that $Min_U^k \leq Min_C^k$ because z_U^k is the global optimal solution to FUM^k . If $Min_U^k < Min_C^k$, then $F^* = Min_U^k$.

That is, we only accept an updated solution for F^* if the solution to the FUM is strictly less than the solution to the FCM at the k^{th} iteration. Similarly, at the $k + 1^{st}$ iteration, the solution for FUM^k is also feasible for FCM^{k+1} because z_U^k satisfies the constraints of FCM^{k+1} by the nature of the problem formulation. Because z_U^k is feasible for FCM^{k+1} , it follows that $Min_C^{k+1} \leq Min_U^k$ since z_C^{k+1} is the global optimal solution to FCM^{k+1} . If $Min_C^{k+1} < Min_U^k$, then $F^* = Min_C^{k+1}$. Therefore, our algorithm produces a set of solutions that are strictly monotone decreasing. In addition, this sequence is bounded below because both Min_C^k and Min_U^k are bounded below. Therefore, our algorithm converges and convergence occurs when $Min_U^k = Min_C^k = F^*$ or $Min_C^{k+1} = Min_U^k = F^*$. More strongly, we know that the algorithm converges finitely because the MIP problems being alternatively solved have discrete and finite feasible regions: the strict monotonicity implies the algorithm must eventually stop. [9] provides details about for convergence theory on monotone decreasing sequences.

6. Computational example. Throughout this paper, we have described the size of the problem that we actually solved for the ACH interiors restructuring. In this section, we provide a small computational example to demonstrate the algorithm and the core decision making capabilities of the model. We will first introduce the inputs for this example and follow with the model runs and solution.

6.1. Inputs. First, we describe the input parameters that affect the dimension of the problem. We have four product programs ($\lambda = 4$) having a total of ten manufacturing requirements ($n = 10$) and four third-party suppliers having seven manufacturing facilities ($m = 7$) to which the product programs need to be sourced. There are thirteen potential processes ($r = 13$), although not all processes are available at every facility. We divide a facility's utilization into ten levels ($\varrho = 10$).

Table 1 lists ten manufacturing requirements \mathcal{C} , each characterized by certain process requirement PR_i . The column PF_i defines the product-program family-numbers; there are four unique product programs. For example, the first two manufacturing requirements correspond to the same product program, so they must be assigned to the same manufacturing facility. The column with $\mathcal{C}^P \subset \mathcal{C}$ provides the indices for the corresponding manufacturing requirements that represent the four unique product programs. Each product program has the following values associated with it: NPV , VV_i , EH_i , UBC_i , IF_i , and CPM_i . Note that $NPV = 3.79$ for every product program. VV_i is the annual production volume, EH_i is the number of required employees, and UBC_i is the unit production cost. IF_i , when non-zero, is the product program's "intellectual property family"; in our example, product programs 1 and 4 fall under the same intellectual property agreement and must be assigned to the same supplier but possibly to different manufacturing facilities of that supplier. Finally, CPM_i , when non-zero, is the product program's "freight-family" identifier. In our

TABLE 1

This table lists the characteristics of four product programs \mathcal{C}^p having ten manufacturing requirements \mathcal{C} .

$i \in \mathcal{C}$	PR_i	PF_i	\mathcal{C}^p	VV_i	EH_i	UBC_i	IF_i	CPM_i
1	149	101	1	43.3	56	\$57.33	105	-
2	13.082	101						
3	72000	102	3	106.3	-	\$40.00	-	100
4	52.1	103	4	19.9	-	\$40.45	105	-
5	230.2	103						
6	456	104	6	24.9	6	\$400.59	-	100
7	125	104						
8	5300	104						
9	6300	104						
10	8174.8	104						

example, product programs 3 and 6, if assigned to the same manufacturing facility, can be shipped for one combined cost.

[Table 2](#) lists seven manufacturing facilities \mathcal{S} , of which facility 4 is a union facility. In this example, we assume that the two Ford facilities must be closed, requiring that all product programs be sourced to the seven listed non-Ford facilities in [Table 2](#): we do not include the two Ford facilities in our set \mathcal{S} . The column U_j denotes which facilities are union facilities. In this example, however, union considerations are set aside; that is, the fewest number of employees required across all union facilities is $UMIN = 0$. The column SU_j provides the supplier identification code: there are three different suppliers. Facilities with the same supplier code are owned by the same supplier. For example, the last two facilities ($j = 6, 7$) with $SU_j = 108$ belong to the same supplier. The facilities each presently have IPP_j types of manufacturing processes; that is, each facility has IPP_j processes having non-zero installed capacity, $IPMC_{pj}$ (see [Table 4](#) for $IPMC_{pj}$ values). We do not incur any closing cost CC_j for not assigning any business to any of the seven facilities because they are all non-Ford facilities; also, all non-Ford facilities are open for business; i.e., $\mathcal{S}^c = \emptyset$.

The underlying assumption of this example is that in the “current state” all product programs are assigned to two company-owned facilities, and we require all production to move to third-party suppliers. Moving production to a different facility often requires moving of tools (e.g. dies) at a certain one-time cost defined at the level of product programs \mathcal{C}^p . We assume that tooling move costs for the product programs are $TC_{1j} = \$0.3$, $TC_{3j} = TC_{4j} = \$0.0$, and $TC_{6j} = \$2.0$ for every $j \in \mathcal{S}$.

Production also requires certain manufacturing “real estate,” such as molding machines or floor space, and there is often a choice of which man-

TABLE 2
This table lists the characteristics of seven manufacturing facilities \mathcal{S} .

$j \in \mathcal{S}$	U_j	SU_j	IPP_j	CC_j
1	-	106	9	-
2	-	107	5	-
3	-	107	6	-
4	1	107	6	-
5	-	107	4	-
6	-	108	7	-
7	-	108	4	-

ufacturing process can be used for a certain manufacturing requirement. Table 3 provides the values for RL_{ip} for each manufacturing requirement on each process. If RL_{ip} is non-zero, then the value in the table represents the amount of process $p \in \mathcal{P}$ needed to support a manufacturing requirement $i \in \mathcal{C}$ if it is fully assigned to that process. For example, if the first requirement is fully assigned to processes 2, then it consumes 149 of that process. Requirements can be spread over multiple processes if available; for example, the first requirement could be split fifty-fifty between processes 3 and 4, creating a load of 74.5 on each process. Note that requirement 1 cannot be assigned to any process other than process 2, 3, or 4.

TABLE 3
The load RL_{ip} caused by manufacturing requirement $i \in \mathcal{C}$ if and when fully assigned to process $p \in \mathcal{P}$.

$i \in \mathcal{C}$	$p \in \mathcal{P}$												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-	149	149	149	-	-	-	-	-	-	-	-	-
2	-	-	-	13.1	13.1	13.1	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	-	72000
4	52.1	52.1	52.1	-	-	-	-	-	-	-	-	-	-
5	-	-	230.2	230.2	230.2	-	-	-	-	-	-	-	-
6	-	-	-	-	-	456	456	456	456	-	-	-	-
7	-	-	-	-	-	-	-	-	-	125	-	-	-
8	-	-	-	-	-	-	-	-	-	-	5300	-	-
9	-	-	-	-	-	-	-	-	-	-	6300	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	8174.8

We know each facility's *installed* process capacity $IPMC_{pj}$ (see Table 4) and the portion SAC_{pj} of that capacity that is *available* for sourcing (see Table 5); the difference $IPMC_{pj} - SAC_{pj}$ represents the capacity promised by that facility to other parties. We assume the value of the capacity

TABLE 4
The total initial installed capacity $IPMC_{pj}$ for process $p \in \mathcal{P}$ at facility $j \in \mathcal{S}$.

$p \in \mathcal{P}$	$j \in \mathcal{S}$						
	1	2	3	4	5	6	7
1	2,400	1,680	-	-	1,440	360	480
2	120	600	-	-	-	480	240
3	-	960	-	120	1,320	120	240
4	1,200	-	120	-	120	240	-
5	150	-	-	-	-	240	-
6	300	-	480	240	-	240	-
7	-	-	-	480	-	-	-
8-9	-	-	-	-	-	-	-
10	150	-	240	120	-	-	-
11	6,500	-	18,252	25,272	-	-	-
12	7,800	480	9,600	12,000	-	-	-
13	40,000	874,000	160,000	178,000	288,000	55,000	14,000

TABLE 5
The available capacity SAC_{pj} of process $p \in \mathcal{P}$ at facility $j \in \mathcal{S}$.

$p \in \mathcal{P}$	$j \in \mathcal{S}$						
	1	2	3	4	5	6	7
1	1,200	359	-	-	1,071	62	-
2	60	241	-	-	-	111	16
3	-	98	-	120	1,047	17	16
4	330	-	-	-	102	168	-
5	50	-	-	-	-	46	-
6	90	-	130	25	-	69	-
7	-	-	-	20	-	-	-
8-9	-	-	-	-	-	-	-
10	80	-	240	-	-	-	-
11	1,000	-	17,002	21,626	-	-	-
12	1,000	400	6,384	8,400	-	-	-
13	10,000	406,010	50,000	75,000	-	16,000	5,000

overload flex factor $COF = 1.0$; i.e., no overload on any of the processes is allowed.

We have the option of moving, if needed, the process capacities from the closed company-owned facilities to a different location, within certain limits and at a certain cost. In this example, we assume that the maximum number of units of capacity of process that can be moved to a facility

$j \in \mathcal{S}$ is as follows: $MPM_{pj} = 3$ for processes $p = 1\text{--}4$, 6 and 8–11; and $MPM_{pj} = 0$ for the remaining processes $p = 5$, 7, 12 and 13. The capacity per unit of process at a facility $j \in \mathcal{S}$ is defined as follows: $CPU_{pj} = 120$ for processes $p = 1\text{--}4$, 6 and 8–10; $CPU_{pj} = 2,880$ for process $p = 11$; and $CPU_{pj} = 0$ for the remaining $p = 5$, 7, 12 and 13.

[Table 6](#) provides the cost for moving a unit of capacity of process to a facility. A table entry with no cost means that you cannot move the specified process to the specified facility. Process moves can only occur in predefined “chunks”; for example, one can only move the stamping machine in its wholeness, even if just a tiny portion of that machine’s capacity would be needed at the new location. Some capacities, such as floor space, cannot be moved at all. In our example, the capacities of processes 5, 7, 12, and 13 are unmoveable; processes 8 and 9 presently have no capacities at any of the facilities (as seen in [Table 5](#)) but can be moved into the facility of our choice if needed.

TABLE 6
The cost MC_{pj} of moving a unit of capacity of process $p \in \mathcal{P}$ to facility $j \in \mathcal{S}$.

$p \in \mathcal{P}$	$j \in \mathcal{S}$						
	1	2	3	4	5	6	7
1	\$0.08	\$0.08	\$0.08	\$0.08	\$0.08	\$0.08	\$0.08
2	\$0.09	\$0.09	\$0.09	\$0.09	\$0.09	\$0.09	\$0.09
3	\$0.10	\$0.10	\$0.10	\$0.10	\$0.10	\$0.10	\$0.10
4	\$0.18	\$0.18	\$0.18	\$0.18	\$0.18	\$0.18	\$0.18
5, 7, 12, 13	-	-	-	-	-	-	-
6	\$0.33	\$0.33	\$0.33	\$0.33	\$0.33	\$0.33	\$0.33
8	\$0.35	\$0.35	\$0.35	\$0.35	\$0.35	\$0.35	\$0.35
9	\$0.38	\$0.38	\$0.38	\$0.38	\$0.38	\$0.38	\$0.38
10	\$0.13	\$0.13	\$0.13	\$0.13	\$0.13	\$0.13	\$0.13
11	\$0.38	\$0.38	\$0.38	\$0.38	\$0.38	\$0.38	\$0.38

Once produced, the goods need to be shipped from the manufacturing facilities to the vehicle assembly plants, at a certain cost. [Table 7](#) provides the net present value freight cost, FC_{ij} , to ship product program $i \in \mathcal{C}^p$ from facility $j \in \mathcal{S}$ to the final vehicle assembly destination. Recall that some parts that are shipped together get a freight discount. Those eligible are shown in [Table 1](#) under the column CPM_i .

Finally, the unit production cost depends on a facility’s utilization. In the first approximation, it is a hyperbolic function accounting for variable and fixed costs; the second-degree factors account for shop floor congestion at critical (high) utilization. The utilization curve has been modeled with ten utilization levels $q \in \mathcal{Q}$. For example, the second utilization level represents facility utilization range from $UB_1 = 10\%$ to $UB_2 = 20\%$. [Table 8](#)

displays the values for the utilization-dependant cost multipliers QC_{qj} for each facility $j \in \mathcal{S}$ in all of the ranges $q \in \mathcal{Q}$.

TABLE 7

Freight cost FC_{ij} , in \$mln, to ship product program $i \in \mathcal{C}^P$ from facility $j \in \mathcal{S}$ to the final vehicle assembly destination. This is the net present value cost over the life of the project, calculated as the annual freight cost multiplied by the NPV factor.

$i \in \mathcal{C}^P$	$j \in \mathcal{S}$						
	1	2	3	4	5	6	7
1	0.190	0.476	1.198	0.941	0.654	0.667	0.689
3	648.772	1,031.588	886.521	1,571.560	224.048	1,161.343	224.048
4	0.324	0.322	0.627	0.067	0.453	0.443	0.464
6	96.819	193.072	259.753	322.666	95.312	224.905	106.048

TABLE 8

This table shows utilization-dependant cost multiplier QC_{qj} for each facility $j \in \mathcal{S}$.

$q \in \mathcal{Q}$	UB_q	$j \in \mathcal{S}$						
		1	2	3	4	5	6	7
1	0.1	1.85	2.57	1.85	1.85	2.57	2.57	1.85
2	0.2	1.4	1.76	1.4	1.4	1.76	1.76	1.4
3	0.3	1.17	1.35	1.17	1.17	1.35	1.35	1.17
4	0.4	1.1	1.22	1.1	1.1	1.22	1.22	1.1
5	0.5	1.061	1.15	1.061	1.061	1.15	1.15	1.061
6	0.6	1.04	1.111	1.04	1.04	1.111	1.111	1.04
7	0.7	1.0201	1.08	1.0201	1.0201	1.08	1.08	1.0201
8	0.8	1.01	1.05	1.01	1.01	1.05	1.05	1.01
9	0.9	1	1.04	1	1	1.04	1.04	1
10	100	1.00001	1.04001	1.00001	1.00001	1.04001	1.04001	1.00001

6.2. Results. We now provide results for the computational example described above, using the algorithm described in Section 5.2.

Iteration 1: Facility capacity model. We first initialize the FUM objective function value, Min_U^0 , to an arbitrarily large value and the utilization levels y_{qj}^0 with the “highest” facility utilization range; that is, we assumed that all facilities were in their maximum utilization range ($q = 10$). Note that by setting all facilities to be at their highest utilization in the first iteration, we are potentially giving them a more beneficial cost multiplier QC_{qj} than they actually deserve. For example, looking at Table 8, if facility 1 is really 25% utilized, then $q = 3$ and $QC_{3,1} = 1.17$, but if we initialize $q = 10 \forall q \in \mathcal{Q}$, then $QC_{10,1} = 1.00001$.

This FCM problem had 124 variables and 71 constraints after the pre-solve. After running the FCM, the manufacturing requirement facility sourcing decisions are in [Table 9](#), the manufacturing process sourcing decisions are in [Table 10](#), and the objective function value $\text{Min}_C^1 = 1362.615864$. Also after running the FCM, we get that the units of capacity to move $m_{6,3} = 3$ (increases the capacity of process 6 at facility 3 from 130 to 490), and $m_{pj} = 0$ for all other $p \in \mathcal{P}$ and $j \in \mathcal{S}$. Note from [Table 3](#) that manufacturing requirement 6 can be run on process 6, 7, 8, or 9 and has a required load of 456. However, we can see in [Table 5](#) that no processes of type 6, 7, 8, or 9 at any of the facilities has enough available capacity to meet the requirement for manufacturing requirement 6. So, the problem would be infeasible if we were not allowed to move units of capacity into a facility. Although the moving costs for the different processes 6, 8, and 9 are the same for facilities 1, 3, 4, and 6 according to [Table 6](#) (process 7 cannot be moved), adding process capacity of type 6 at facilities 1, 4, and 6 and process capacity of type 8 and 9 at all facilities would require moving 4 units of process capacity: moving process capacity of type 6 to facility 3 only requires moving 3 units of process capacity.

TABLE 9

This table shows the output value x_{ij} specifying whether or not manufacturing requirement $i \in \mathcal{C}$ is produced by facility $j \in \mathcal{S}$. These output values are equivalent for the first and the second executions of both the FCM and the FUM.

$i \in \mathcal{C}$	$j \in \mathcal{S}$						
	1	2	3	4	5	6	7
1–2	1	-	-	-	-	-	-
3	-	1	-	-	-	-	-
4–5	1	-	-	-	-	-	-
6–10	-	-	1	-	-	-	-

Since we are solving the FCM in the first iteration, and Min_U^0 is set to some arbitrarily large number, we do not need to check for convergence. We proceed to the next step, where we calculate the values for npp_j as defined in the algorithm section, pmc_{pj}^k as defined in (3.14), and ipu_j^k as defined in (3.18). Note that although we move $m_{6,3} = 3$ units of capacity of process $p = 6$ to facility $j = 3$, some units of capacity of process six were already available at facility three, so all the values of npp_j as defined in [Table 2](#) are the same. We update the other two values for the affected facility number $j = 3$: $pmc_{6,3}^1 = 480 + 3 \cdot 120 = 840$, and $ipu_3^1 = (\frac{120-0}{120} + \frac{840-130}{840} + \frac{240-240}{240} + \frac{18252-17002}{18252} + \frac{9600-6384}{9600} + \frac{160000-50000}{160000})/6 = 0.7680378$.

TABLE 10

This table shows the output from the first and the second executions of the FCM: the actual load $\sum_{j \in \mathcal{S}, q \in \mathcal{Q}} RL_{ip} v_{ipqj}$ of manufacturing requirement $i \in \mathcal{C}$ on process $p \in \mathcal{P}$.

$i \in \mathcal{C}$	$p \in \mathcal{P}$												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-	-	-	149	-	-	-	-	-	-	-	-	-
2	-	-	-	-	-	13.1	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	72,000.0	
4	52.1	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	-	181	49.2	-	-	-	-	-	-	-	-
6	-	-	-	-	-	456	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	125	-	-	-	-
8	-	-	-	-	-	-	-	-	-	5300	-	-	-
9	-	-	-	-	-	-	-	-	-	-	6300	-	-
10	-	-	-	-	-	-	-	-	-	-	-	8,174.8	

Iteration 1: Facility utilization model. The output from iteration 1 of the FCM was then used as the input to the FUM: so, in this run, facility 3's available capacity of process 6 was increased by three units.

This FUM problem had 151 variables and 113 constraints after the pre-solve. After running the FUM, we get the following outputs: objective function value, facility utilization, manufacturing requirement sourcing decisions regarding facilities and processes. The objective function value is $Min_U^1 = 1364.143179$. Note that the objective function value for the FUM in iteration 1 is greater than the objective function value for the FCM in iteration 1 (i.e. $Min_U^1 > Min_C^1$). This is because the FCM got a better utilization cost multiplier than was actually feasible, while the utilization level constraints were enforced in the FUM, giving the accurate, yet higher, utilization cost multiplier.

The manufacturing requirements are sourced to the same facilities as in iteration 1 of the FCM (see Table 9). The facilities have the following utilization levels: facility 1 is utilized at 70–80% ($y_{8,1} = 1$); facilities 2 and 3 are utilized at 60–70% ($y_{7,2} = y_{7,3} = 1$); and all the remaining facility utilization levels, y_{qj} , are zeroes. Recall that utilization level is only defined for the facilities to which we source a share of our own business. Hence, we use the the post-processing technique in Section 5.2.2 to re-establish the utilization levels of non-Ford utilized facilities 4, ..., 7 to be the same as their initial utilization levels (see Table 12): this results in $y_{q_j,j}$.

The manufacturing requirement process sourcing decisions are in Table 11. It is instructive to compare these results against the required loads RL_{ip} listed in Table 3. For example, the manufacturing requirement num-

ber 1 (requiring a load of 149 on any or several of the processes 2, 3, or 4) has been split between two processes: a load of 17.9 on process 2 and a load of 131.1 on process 4. Note that the FUM tends to source one manufacturing requirement to multiple processes if cost effective because a facility's utilization is actually an average of the process utilization at that facility.

Before proceeding to the second iteration, we apply the method in Section 5.2.2 to update the values of cost multipliers $QC_{\bar{q}_j j}^k$ (see Table 12), requirement costs UC_{ij}^k , and product program costs PC_{ij}^k .

TABLE 11

This table shows the output from the first execution of the FUM: the actual load $\sum_{j \in \mathcal{S}, q \in \mathcal{Q}} RL_{ip} v_{ipqj}$ of manufacturing requirement $i \in \mathcal{C}$ on process $p \in \mathcal{P}$.

$i \in \mathcal{C}$	$p \in \mathcal{P}$												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-	17.9	-	131.1	-	-	-	-	-	-	-	-	-
2	-	-	-	-	-	13.1	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	-	72000
4	52.1	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	-	180.2	50	-	-	-	-	-	-	-	-
6	-	-	-	-	-	456	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	125	-	-	-
8	-	-	-	-	-	-	-	-	-	-	5300	-	-
9	-	-	-	-	-	-	-	-	-	-	-	6300	-
10	-	-	-	-	-	-	-	-	-	-	-	-	8174.8

TABLE 12

This table shows the utilization ranges \bar{q}_j for all facilities, as calculated by the first execution of FUM (for Ford-utilized facilities 1 ... 3) and enhanced by post-processing logic of Section 5.2.2 (for non-Ford utilized facilities 4 ... 7). The corresponding cost multipliers $QC_{\bar{q}_j j}^k$ are then passed to the second execution of the FCM.

	$j \in \mathcal{S}$						
	1	2	3	4	5	6	7
\bar{q}_j	8	7	7	5	5	8	9
$QC_{\bar{q}_j j}^k$	1.01	1.08	1.0201	1.061	1.15	1.05	1

Iteration 2: Facility capacity model. In the second iteration, we use the utilization levels from the FUM in the previous iteration as inputs. The objective function value from the run of this FCM is $Min_C^2 = 1364.143179$, which is the same as the objective function value in the FUM in the first iteration. However, the manufacturing requirement

facility sourcing decisions and manufacturing process sourcing decisions are the same as in the run of the first FCM, see Tables 9 and 10, respectively. In the FCM, the overall facility has the same utilization as in the FUM, but the average process utilization between the two solutions is different. Since the objective function value from the run of this second FCM is the same as the objective value from the run of the first FUM, the algorithm has converged.

7. Summary. In this paper, we presented a real world problem that uses operations research techniques to aid in restructuring Ford's interiors business in a distressed supplier environment. We first gave an overview of the business problem and the importance behind using a data-driven approach to arrive at sourcing decisions in an area of strategic importance to the company.

The underlying model that described the business problem of identifying feasible allocations of 229 manufacturing operations for 42 product programs at more than 50 different supplier sites is a large scale MINLP. We described how we reformulated the large scale MINLP by linearizing the objective function through introduction of discrete utilization levels. We then decoupled the discrete MINLP into two easily solved MIPs for which solution techniques are well known, and we developed an algorithm that iteratively solved the MIPs until convergence. We provided a decision support system that allowed the users to easily perform scenario and sensitivity analysis. This feature was especially important in practice, since considerations other than cost (e.g. union relations, intellectual property, concentration of sourcing) may play a part in evaluating alternative sourcing decisions.

As a result of the availability of the new tool, a new process was developed by which ACH and Purchasing identified optimal and next-best suppliers for each product program in the portfolio, allowing targeted Requests For Quotes (RFQ) to these top suppliers. This practice represents a major departure from standard procedures (typically all suppliers would be sent RFQs).

In the final (May, 2007) approved strategy based on the results of the tool output and resulting targeted market tests, 39 of 42 product programs agreed completely with the optimal recommendation produced by the tool. Perhaps most importantly, the tools output was used as a direct basis for securing \$45 million in restructuring funding to achieve the recommended sourcing pattern, thereby saving the company approximately \$50-55 million in additional costs over a five-year period.

Although the tool was developed for the problem of re-sourcing plastic interiors, it can easily be adapted to address similar strategic sourcing issues for other classes of commodities. We provided a small computational example to illustrate the use and functionality of the algorithm and solution approach.

Acknowledgements. We would like to acknowledge the key involvement of our business partners Chip McDaniel, formerly of Ford, and Mike Wolcott of ACH for posing this problem, helping shape the decisions of the model, painstakingly gathering all of the data, and actively engaging in testing and validation of the model.

REFERENCES

- [1] C. FLOUDAS, *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*, Oxford University Press, Inc., 1995.
- [2] J. NOCEDAL AND S. WRIGHT, *Numerical Optimization*, Springer, New York, NY, 1999.
- [3] ILOG, *ILOG OPL-CPLEX Development System*, <http://www.ilog.com/products/oplstudio/>, 2008.
- [4] FORD MOTOR COMPANY, 2006. *Ford Accelerates “Way Forward.”* http://media.ford.com/article_display.cfm?article_id=24261. Retrieved June 23, 2009.
- [5] I. NOWAK, *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*, Birkhäuser Verlag, 2005.
- [6] L. WOLSEY, *Integer Programming*, John Wiley & Sons, Inc., 1998.
- [7] E. KLAMPFL, Y. FRADKIN, C. McDANIEL, AND M. WOLCOTT, *Ford Optimizes Urgent Sourcing Decisions in a Distressed Supplier Environment*, Interfaces, Special Section: The Daniel H. Wagner Prize for Excellence in Operations Research Practice, Editor: Joseph H. Disenza, September-October 2009.
- [8] R. RARDIN, 2000. *Optimization in Operations Research*. Prentice Hall, Inc., Upper Saddle River, NJ.
- [9] J. DESHPANDE, 2004. *Mathematical Analysis and Applications: An Introduction*. Alpha Science Int'l Ltd., Harrow, UK.
- [10] SHARON SILKE CARTY, USA TODAY, March 27, 2008. *Many auto parts suppliers failed to widen base* http://www.usatoday.com/money/industries/manufacturing/2008-03-26-auto-parts-suppliers_N.htm. Retrieved November 20, 2008.
- [11] AUTOMOTIVEWORLD.COM, February 14, 2008. *US: Blue Water files for bankruptcy* <http://www.automotiveworld.com/news/powertrain/66356-us-blue-water-files-for-bankruptcy>. Retrieved November 20, 2008.

A BENCHMARK LIBRARY OF MIXED-INTEGER OPTIMAL CONTROL PROBLEMS

SEBASTIAN SAGER*

Abstract. Numerical algorithm developers need standardized test instances for empirical studies and proofs of concept. There are several libraries available for finite-dimensional optimization, such as the `netlib` or the `miplib`. However, for mixed-integer optimal control problems (MIOCP) this is not yet the case. One explanation for this is the fact that no dominant standard format has been established yet. In many cases instances are used in a discretized form, but without proper descriptions on the modeling assumptions and discretizations that have been applied. In many publications crucial values, such as initial values, parameters, or a concise definition of all constraints are missing.

In this contribution we intend to establish the basis for a benchmark library of mixed-integer optimal control problems that is meant to be continuously extended online on the open community web page <http://mintoc.de>. The guiding principles will be comprehensiveness, a detailed description of where a model comes from and what the underlying assumptions are, a clear distinction between problem and method description (such as a discretization in space or time), reproducibility of solutions and a standardized problem formulation. Also, the problems will be classified according to model and solution characteristics. We do not benchmark MIOCP solvers, but provide a library infrastructure and sample problems as a basis for future studies.

A second objective is to formulate mixed-integer nonlinear programs (MINLPs) originating from these MIOCPs. The snag is of course that we need to apply one out of several possible method-specific discretizations in time and space in the first place to obtain a MINLP. Yet the resulting MINLPs originating from control problems with an indication of the currently best known solution are hopefully a valuable test set for developers of generic MINLP solvers. The problem specifications can also be downloaded from <http://mintoc.de>.

AMS(MOS) subject classifications. Primary 1234, 5678, 9101112.

1. Introduction. For empirical studies and proofs of concept, developers of optimization algorithms need standardized test instances. There are several libraries available, such as the `netlib` for linear programming (LP) [4], the Schittkowski library for nonlinear programming (NLP) [59], the `miplib` [44] for mixed-integer linear programming (MILP), or more recently the `MINLPLib` [13] and the CMU-IBM Cyber-Infrastructure for mixed-integer nonlinear programming (MINLP) collaborative site [15]. Further test libraries and related links can be found on [12], a comprehensive testing environment is *CUTER* [28]. The solution of these problems with different solvers is facilitated by the fact that standard formats such as the *standard input format* (SIF) or the *Mathematical Programming System* format (MPS) have been defined.

Collections of optimal control problems (OCPs) in ordinary differential equations (ODE) and in differential algebraic equations (DAE) have also

*Interdisciplinary Center for Scientific Computing, University of Heidelberg, 69120 Heidelberg, Germany.

been set up. The PROPT (a matlab toolkit for dynamic optimization using collocation) homepage states over 100 test cases from different applications with their results and computation time, [32]. With the software package dsoa [20] come currently 77 test problems. The ESA provides a test set of global optimization spacecraft trajectory problems and their best putative solutions [3].

This is a good starting point. However, no standard has evolved yet as in the case of finite-dimensional optimization. The specific formats for which only few optimization / optimal control codes have an interface, insufficient information on the modeling assumptions, or missing initial values, parameters, or a concise definition of all constraints make a transfer to different solvers and environments very cumbersome. The same is true for hybrid systems, which incorporate MIOCPs as defined in this paper as a special case. Two benchmark problems have been defined at [19].

Although a general open library would be highly desirable for optimal control problems, we restrict ourselves here to the case of MIOCPs, in which some or all of the control values and functions need to take values from a finite set. MIOCPs are of course more general than OCPs as they include OCPs as a special case, however the focus in this library will be on integer aspects. We want to be general in our formulation, without becoming too abstract. It will allow to incorporate ordinary and partial differential equations, as well as algebraic constraints. Most hybrid systems can be formulated by means of state-dependent switches. Closed-loop control problems are on a different level, because a unique and comparable scenario would include well-defined external disturbances. We try to leave our approach open to future extensions to nonlinear model predictive control (NMPC) problems, but do not incorporate them yet. The formulation allows for different kinds of objective functions, e.g., time minimal or of tracking type, and of boundary constraints, e.g., periodicity constraints. Abstract problem formulations, together with a proposed categorization of problems according to model, objective, and solution characteristics will be given in Section 2.

MIOCPs include features related to different mathematical disciplines. Hence, it is not surprising that very different approaches have been proposed to analyze and solve them. There are three generic approaches to solve model-based optimal control problems, compare [8]: first, solution of the *Hamilton-Jacobi-Bellman equation* and in a discrete setting *Dynamic Programming*, second *indirect methods*, also known as the *first optimize, then discretize* approach, and third *direct methods (first optimize, then discretize)* and in particular *all-at-once approaches* that solve the simulation and the optimization task simultaneously. The combination with the additional combinatorial restrictions on control functions comes at different levels: for free in dynamic programming, as the control space is evaluated anyhow, by means of an enumeration in the inner optimization problem of

the necessary conditions of optimality in Pontryagin's maximum principle, or by various methods from integer programming in the direct methods.

Even in the case of direct methods, there are multiple alternatives to proceed. Various approaches have been proposed to discretize the differential equations by means of shooting methods or collocation, e.g., [10, 7], to use global optimization methods by under- and overestimators, e.g., [18, 48, 14], to optimize the time-points for a given switching structure, e.g., [36, 26, 58], to consider a static optimization problem instead of the transient behavior, e.g., [30], to approximate nonlinearities by piecewise-linear functions, e.g., [45], or by approximating the combinatorial decisions by continuous formulations, as in [11] for drinking water networks. Also problem (re)formulations play an important role, e.g., outer convexification of nonlinear MIOCPs [58], the modeling of MPECs and MPCCs [6, 5], or mixed-logic problem formulations leading to disjunctive programming, [50, 29, 47].

We do not want to discuss reformulations, solvers, or methods in detail, but rather refer to [58, 54, 29, 5, 47, 50] for more comprehensive surveys and further references. The main purpose of mentioning them is to point out that they all discretize the optimization problem in function space in a different manner, and hence result in different mathematical problems that are actually solved on a computer.

We have two objectives. First, we intend to establish the basis for a benchmark library of mixed-integer optimal control problems that is meant to be continuously extended online on the open community web page <http://mintoc.de>. The guiding principles will be comprehensiveness, a detailed description of where a model comes from and what the underlying assumptions are, a clear distinction between problem and method description (such as a discretization in space or time), reproducibility of solutions and a standardized problem formulation that allows for an easy transfer, once a method for discretization has been specified, to formats such as AMPL or GAMS. Also, the problems will be classified according to model and solution characteristics.

Although the focus of this paper is on formulating MIOCPs before any irreversible reformulation and numerical solution strategy has been applied, a second objective is to provide specific MINLP formulations as benchmarks for developers of MINLP solvers. Powerful commercial MILP solvers and advances in MINLP solvers as described in the other contributions to this book make the usage of general purpose MILP/MINLP solvers more and more attractive. Please be aware however that the MINLP formulations we provide in this paper are only one out of many possible ways to formulate the underlying MIOCP problems.

In Section 2 a classification of problems is proposed. Sections 3 to 11 describe the respective control problems and currently best known solutions. In Section 12 two specific MINLP formulations are presented for illustration. Section 13 gives a conclusion and an outlook.

2. Classifications. The MIOCPs in our benchmark library have different characteristics. In this section we describe these general characteristics, so we can simply list them later on where appropriate. Beside its origins from application fields such as mechanical engineering, aeronautics, transport, systems biology, chemical engineering and the like, we propose three levels to characterize a control problem. First, characteristics of the model from a mathematical point of view, second the formulation of the optimization problem, and third characteristics of an optimal solution from a control theory point of view. We will address these three in the following subsections.

Although we strive for a standardized problem formulation, we do not formulate a specific generic formulation as such. Such a formulation is not even agreed upon for PDEs, let alone the possible extensions in the direction of algebraic variables, network topologies, logical connections, multi-stage processes, MPEC constraints, multiple objectives, functions including higher-order derivatives and much more that might come in. Therefore we chose to start with a very abstract formulation, formulate every control problem in its specific way as is adequate and to connect the two by using a characterization. On the most abstract level, we want to solve an optimization problem that can be written as

$$\begin{aligned} \min_{x,u,v} \quad & \Phi[x, u, v] \\ \text{s.t.} \quad & 0 = F[x, u, v], \\ & 0 \leq C[x, u, v], \\ & 0 = \Gamma[x]. \end{aligned} \tag{2.1}$$

Here $x(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}^{n_x}$ denotes the differential-algebraic states¹ in a d -dimensional space. Until now, for most applications we have $d = 1$ and the independent variable time $t \in [t_0, t_f]$, the case of ordinary or algebraic differential equations. $u(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}^{n_u}$ and $v(\cdot) : \mathbb{R}^d \mapsto \Omega$ are controls, where $u(\cdot)$ are continuous values that map to \mathbb{R}^{n_u} , and $v(\cdot)$ are controls that map to a finite set Ω . We allow also constant-in-time or constant-in-space control values rather than distributed controls.

We will also use the term *integer control* for $v(\cdot)$, while *binary control* refers to $\omega(t) \in \{0, 1\}^{n_\omega}$ that will be introduced later. We use the expression *relaxed*, whenever a restriction $v(\cdot) \in \Omega$ is relaxed to a convex control set, which is typically the convex hull, $v(\cdot) \in \text{conv}\Omega$.

Basically two different kinds of switching events are at the origin of hybrid systems, controllable and state-dependent ones. The first kind is due to degrees of freedom for the optimization, in particular with controls that may only take values from a finite set. The second kind is due to

¹Note that we use the notation common in control theory with x as differential states and u as controls, not the PDE formulation with x as independent variable and u as differential states.

state-dependent switches in the model equations, e.g., ground contact of a robot leg or overflow of weirs in a distillation column. The focus in the benchmark library is on the first kind of switches, whereas the second one is of course important for a classification of the model equations, as for certain MIOCPs both kinds occur.

The model equations are described by the functional $F[\cdot]$, to be specified in Section 2.1. The objective functional $\Phi[\cdot]$, the constraints $C[\cdot]$ that may include control- and path-constraints, and the interior point constraints $\Gamma[x]$ that specify also the boundary conditions are classified in Section 2.2. In Section 2.3 characteristics of an optimal solution from a control theory point of view are listed.

The formulation of optimization problems is typically not unique. Sometimes, as in the case of MPEC reformulations of state-dependent switches [5], disjunctive programming [29], or outer convexification [58], reformulations may be seen as part of the solution approach in the sense of the *modeling for optimization paradigm* [47]. Even in obvious cases, such as a Mayer term versus a Lagrange term formulation, they may be mathematically, but not necessarily algorithmically equivalent. We propose to use either the original or the most adequate formulation of the optimization problem and list possible reformulations as variants.

2.1. Model classification. This Section addresses possible realizations of the state equation

$$0 = F[x, u, v]. \quad (2.2)$$

We assume throughout that the differential-algebraic states x are uniquely determined for appropriate boundary conditions and fixed (u, v) .

2.1.1. ODE model. This category includes all problems constrained by the solution of explicit ordinary differential equations (ODE). In particular, no algebraic variables and derivatives with respect to one independent variable only (typically time) are present in the mathematical model. Equation (2.2) reads as

$$\dot{x}(t) = f(x(t), u(t), v(t)), \quad t \in [0, t_f], \quad (2.3)$$

for $t \in [t_0, t_f]$ almost everywhere. We will often leave the argument (t) away for notational convenience.

2.1.2. DAE model. If the model includes algebraic constraints and variables, for example from conservation laws, a problem will be categorized as a DAE model. Equality (2.2) will then include both differential equations and algebraic constraints that determine the algebraic states in dependence of the differential states and the controls. A more detailed classification includes the index of the algebraic equations.

2.1.3. PDE model. If $d > 1$ the model equation (2.2) becomes a partial differential equation (PDE). Depending on whether convection or diffusion prevails, a further classification into hyperbolic, elliptic, or parabolic equations is necessary. A more elaborate classification will evolve as more PDE constrained MIOCPs are described on <http://mintoc.de>. In this work one PDE-based instance is presented in Section 11.

2.1.4. Outer convexification. For time-dependent and space-independent integer controls often another formulation is beneficial, e.g., [37]. For every element v^i of Ω a binary control function $\omega_i(\cdot)$ is introduced. Equation (2.2) can then be written as

$$0 = \sum_{i=1}^{n_\omega} F[x, u, v^i] \omega_i(t), \quad t \in [0, t_f]. \quad (2.4)$$

If we impose the special ordered set type one condition

$$\sum_{i=1}^{n_\omega} \omega_i(t) = 1, \quad t \in [0, t_f], \quad (2.5)$$

there is a bijection between every feasible integer function $v(\cdot) \in \Omega$ and an appropriately chosen binary function $\omega(\cdot) \in \{0, 1\}^{n_\omega}$, compare [58]. The relaxation of $\omega(t) \in \{0, 1\}^{n_\omega}$ is given by $\omega(t) \in [0, 1]^{n_\omega}$. We will refer to (2.4) and (2.5) as *outer convexification* of (2.2). This characteristic applies to the control problems in Sections 3, 6, 9, 10, and 11.

2.1.5. State-dependent switches. Many processes are modelled by means of state-dependent switches that indicate, e.g., model changes due to a sudden ground contact of a foot or a weir overflow in a chemical process. Mathematically, we write

$$0 = F_i[x, u, v] \quad \text{if } \sigma_i(x(t)) \geq 0 \quad (2.6)$$

with well defined switching functions $\sigma_i(\cdot)$ for $t \in [0, t_f]$. This characteristic applies to the control problems in Sections 6 and 8.

2.1.6. Boolean variables. Discrete switching events can also be expressed by means of Boolean variables and logical implications. E.g., by introducing logical functions $\delta_i : [0, t_f] \mapsto \{\text{true}, \text{false}\}$ that indicate whether a model formulation $F_i[x, u, v]$ is active at time t , both state-dependent switches and outer convexification formulations may be written as *disjunctive programs*, i.e., optimization problems involving Boolean variables and logical conditions. Using disjunctive programs can be seen as a more natural way of modeling discrete events and has the main advantage of resulting in tighter relaxations of the discrete decisions, when compared to integer programming techniques. More details can be found in [29, 46, 47].

2.1.7. Multistage processes. Processes of interest are often modelled as multistage processes. At transition times the model can change, sometimes in connection with a state-dependent switch. The equations read as

$$0 = F_i[x, u, v] \quad t \in [t_i, t_{i+1}] \quad (2.7)$$

on a time grid $\{t_i\}_i$. With smooth transfer functions also changes in the dimension of optimization variables can be incorporated, [43].

2.1.8. Unstable dynamics. For numerical reasons it is interesting to keep track of instabilities in process models. As small changes in inputs lead to large changes in outputs, challenges for optimization methods arise. This characteristic applies to the control problems in Sections 3 and 7.

2.1.9. Network topology. Complex processes often involve an underlying network topology, such as in the control of gas or water networks [45, 11]. The arising structures should be exploited by efficient algorithms.

2.2. Classification of the optimization problem. The optimization problem (2.1) is described by means of an objective functional $\Phi[\cdot]$ and inequality constraints $C[\cdot]$ and equality constraints $\Gamma[\cdot]$. The constraints come in form of multipoint constraints that are defined on a time grid $t_0 \leq t_1 \leq \dots \leq t_m = t_f$, and of path-constraints that need to hold almost everywhere on the time horizon. The equality constraints $\Gamma[\cdot]$ will often fix the initial values or impose a periodicity constraint. In this classification we assume all functions to be sufficiently often differentiable.

In the future, the classification will also include problems with non-differentiable objective functions, multiple objectives, online control tasks including feedback, indication of nonconvexities, and more characteristics that allow for a specific choice of test instances.

2.2.1. Minimum time. This is a category with all control problems that seek for time-optimal solutions, e.g., reaching a certain goal or completing a certain process as fast as possible. The objective function is of Mayer type, $\Phi[\cdot] = t_f$. This characteristic applies to the control problems in Sections 3, 9, and 10.

2.2.2. Minimum energy. This is a category with all control problems that seek for energy-optimal solutions, e.g., reaching a certain goal or completing a certain process with a minimum amount of energy. The objective function is of Lagrange type and sometimes proportional to a minimization of the squared control (e.g., acceleration) $u(\cdot)$, e.g., $\Phi[\cdot] = \int_{t_0}^{t_f} u^2 dt$. Almost always an upper bound on the free end time t_f needs to be specified. This characteristic applies to the control problems in Sections 6 and 8.

2.2.3. Tracking problem. This category lists all control problems in which a tracking type Lagrange functional of the form

$$\Phi[\cdot] = \int_{t_0}^{t_f} \|x(\tau) - x^{\text{ref}}\|_2^2 \, d\tau \quad (2.8)$$

is to be minimized. This characteristic applies to the control problems in Sections 4, 5, and 7.

2.2.4. Periodic processes. This is a category with all control problems that seek periodic solutions, i.e., a condition of the kind

$$\Gamma[x] = P(x(t_f)) - x(t_0) = 0, \quad (2.9)$$

has to hold. $P(\cdot)$ is an operation that allows, e.g., for a perturbation of states (such as needed for the formulation of Simulated Moving Bed processes, Section 11, or for offsets of angles by a multiple of 2π such as in driving on closed tracks, Section 10). This characteristic applies to the control problems in Sections 8, 10, and 11.

2.2.5. Equilibrium constraints. This category contains mathematical programs with equilibrium constraints (MPECs). An MPEC is an optimization problem constrained by a variational inequality, which takes for generic variables / functions y_1, y_2 the following general form:

$$\begin{aligned} \min_{y_1, y_2} \quad & \Phi(y_1, y_2) \\ \text{s.t.} \quad & 0 = F(y_1, y_2), \\ & 0 \leq C(y_1, y_2), \\ & 0 \leq (\mu - y_2)^T \phi(y_1, y_2), \quad y_2 \in Y(y_1), \quad \forall \mu \in Y(y_1) \end{aligned} \quad (2.10)$$

where $Y(y_1)$ is the feasible region for the variational inequality and given function $\phi(\cdot)$. Variational inequalities arise in many domains and are generally referred to as equilibrium constraints. The variables y_1 and y_2 may be controls or states.

2.2.6. Complementarity constraints. This category contains optimization problems with complementarity constraints (MPCCs), for generic variables / functions y_1, y_2, y_3 in the form of

$$\begin{aligned} \min_{y_1, y_2, y_3} \quad & \Phi(y_1, y_2, y_3) \\ \text{s.t.} \quad & 0 = F(y_1, y_2, y_3), \\ & 0 \leq C(y_1, y_2, y_3), \\ & 0 \leq y_1 \perp y_2 \geq 0. \end{aligned} \quad (2.11)$$

The complementarity operator \perp implies the disjunctive behavior

$$y_{1,i} = 0 \quad \text{OR} \quad y_{2,i} = 0 \quad \forall i = 1 \dots n_y.$$

MPCCs may arise from a reformulation of a bilevel optimization problem by writing the optimality conditions of the inner problem as variational constraints of the outer optimization problem, or from a special treatment of state-dependent switches, [5]. Note that all MPCCs can be reformulated as MPECs.

2.2.7. Vanishing constraints. This category contains mathematical programs with vanishing constraints (MPVCs). The problem

$$\begin{aligned} \min_y \quad & \Phi(y) \\ \text{s.t.} \quad & 0 \geq g_i(y)h_i(y), \quad i \in \{1, \dots, m\} \\ & 0 \leq h(y) \end{aligned} \tag{2.12}$$

with smooth functions $g, h : \mathbb{R}^{n_y} \mapsto \mathbb{R}^m$ is called MPVC. Note that every MPVC can be transformed into an MPEC [2, 33]. Examples for vanishing constraints are engine speed constraints that are only active if the corresponding gear control is nonzero. This characteristic applies to the control problems in Sections 9, and 10.

2.3. Solution classification. The classification that we propose for switching decisions is based on insight from Pontryagin's maximum principle, [49], applied here only to the relaxation of the binary control functions $\omega(\cdot)$, denoted by $\alpha(\cdot) \in [0, 1]^{n_\omega}$. In the analysis of linear control problems one distinguishes three cases: bang-bang arcs, sensitivity-seeking arcs, and path-constrained arcs, [61], where an arc is defined to be a nonzero time-interval. Of course a problem's solution can show two or even all three behaviors at once on different time arcs.

2.3.1. Bang-bang arcs. Bang-bang arcs are time intervals on which the control bounds are active, i.e., $\alpha_i(t) \in \{0, 1\} \forall t$. The case where the optimal solution contains only bang-bang arcs is in a sense the easiest. The solution of the relaxed MIOCP will be integer feasible, if the control discretization grid is a superset of the switching points of the optimal control. Hence, the main goal will be to adapt the control discretization grid such that the solution of the relaxed problem is already integer. Also on fixed time grids good solutions are easy to come up with, as rounded solutions approximate the integrated difference between relaxed and binary solution very well.

A prominent example of this class is time-optimal car driving, see Section 9 and see Section 10. Further examples of “bang-bang solutions” include free switching of ports in Simulated Moving Bed processes, see Section 11, unconstrained energy-optimal operation of subway trains see Section 6, a simple F-8 flight control problem see Section 3, and phase resetting in biological systems, such as in Section 7.

2.3.2. Path-constrained arcs. Whenever a path constraint is active, i.e., it holds $c_i(x(t)) = 0 \forall t \in [t^{\text{start}}, t^{\text{end}}] \subseteq [0, t_f]$, and no continuous

control $u(\cdot)$ can be determined to compensate for the changes in $x(\cdot)$, naturally $\alpha(\cdot)$ needs to do so by taking values in the interior of its feasible domain. An illustrating example has been given in [58], where velocity limitations for the energy-optimal operation of New York subway trains are taken into account, see Section 6. The optimal integer solution does only exist in the limit case of infinite switching (Zeno behavior), or when a tolerance is given. Another example is compressor control in supermarket refrigeration systems, see Section 8. Note that all applications may comprise path-constrained arcs, once path constraints need to be added.

2.3.3. Sensitivity-seeking arcs. We define sensitivity-seeking (also compromise-seeking) arcs in the sense of Srinivasan and Bonvin, [61], as arcs which are neither bang-bang nor path-constrained and for which the optimal control can be determined by time derivatives of the Hamiltonian. For control-affine systems this implies so-called singular arcs.

A classical small-sized benchmark problem for a sensitivity-seeking (singular) arc is the Lotka-Volterra Fishing problem, see Section 4. The treatment of sensitivity-seeking arcs is very similar to the one of path-constrained arcs. As above, an approximation up to any a priori specified tolerance is possible, probably at the price of frequent switching.

2.3.4. Chattering arcs. Chattering controls are bang-bang controls that switch infinitely often in a finite time interval $[0, t_f]$. An extensive analytical investigation of this phenomenon can be found in [63]. An example for a chattering arc solution is the famous example of Fuller, see Section 5.

2.3.5. Sliding mode. Solutions of model equations with state-dependent switches as in (2.6) may show a sliding mode behavior in the sense of Filippov systems [21]. This means that at least one of the functions $\sigma_i(\cdot)$ has infinitely many zeros on the finite time interval $[0, t_f]$. In other words, the right hand side switches infinitely often in a finite time horizon.

The two examples with state-dependent switches in this paper in Sections 6 and 8 do not show sliding mode behavior.

3. F-8 flight control. The F-8 aircraft control problem is based on a very simple aircraft model. The control problem was introduced by Kaya and Noakes [36] and aims at controlling an aircraft in a time-optimal way from an initial state to a terminal state. The mathematical equations form a small-scale ODE model. The interior point equality conditions fix both initial and terminal values of the differential states. The optimal, relaxed control function shows bang-bang behavior. The problem is furthermore interesting as it should be reformulated equivalently. Despite the reformulation the problem is nonconvex and exhibits multiple local minima.

3.1. Model and optimal control problem. The F-8 aircraft control problem is based on a very simple aircraft model in ordinary differential equations, introduced by Garrard [24]. The differential states consist of x_0 as the angle of attack in radians, x_1 as the pitch angle, and x_2 as the pitch

rate in rad/s. The only control function $w = w(t)$ is the tail deflection angle in radians. The control objective is to control the airplane from one point in space to another in minimum time. For $t \in [0, T]$ almost everywhere the mixed-integer optimal control problem is given by

$$\begin{aligned} & \min_{x,w,T} && T \\ \text{s.t. } & \dot{x}_0 = && -0.877 x_0 + x_2 - 0.088 x_0 x_2 + 0.47 x_0^2 - 0.019 x_1^2 \\ & && - x_0^2 x_2 + 3.846 x_0^3 \\ & && - 0.215 w + 0.28 x_0^2 w + 0.47 x_0 w^2 + 0.63 w^3 \\ & \dot{x}_1 = x_2 \\ & \dot{x}_2 = && -4.208 x_0 - 0.396 x_2 - 0.47 x_0^2 - 3.564 x_0^3 \\ & && - 20.967 w + 6.265 x_0^2 w + 46 x_0 w^2 + 61.4 w^3 \\ & x(0) = (0.4655, 0, 0)^T, && x(T) = (0, 0, 0)^T, \\ & w(t) \in \{-0.05236, 0.05236\}, && t \in [0, T]. \end{aligned} \tag{3.1}$$

In the control problem, both initial and terminal values of the differential states are fixed. The control $w(t)$ is restricted to take values from a finite set only. Hence, the control problem can be reformulated equivalently to

$$\begin{aligned} & \min_{x,w,T} && T \\ \text{s.t. } & \dot{x}_0 = && -0.877 x_0 + x_2 - 0.088 x_0 x_2 + 0.47 x_0^2 - 0.019 x_1^2 \\ & && - x_0^2 x_2 + 3.846 x_0^3 \\ & && + 0.215 \xi - 0.28 x_0^2 \xi + 0.47 x_0 \xi^2 - 0.63 \xi^3 \\ & && - (0.215 \xi - 0.28 x_0^2 \xi - 0.63 \xi^3) 2w \\ & \dot{x}_1 = x_2 \\ & \dot{x}_2 = && -4.208 x_0 - 0.396 x_2 - 0.47 x_0^2 - 3.564 x_0^3 \\ & && + 20.967 \xi - 6.265 x_0^2 \xi + 46 x_0 \xi^2 - 61.4 \xi^3 \\ & && - (20.967 \xi - 6.265 x_0^2 \xi - 61.4 \xi^3) 2w \\ & x(0) = (0.4655, 0, 0)^T, && x(T) = (0, 0, 0)^T, \\ & w(t) \in \{0, 1\}, && t \in [0, T] \end{aligned} \tag{3.2}$$

with $\xi = 0.05236$. Note that there is a bijection between optimal solutions of the two problems, and that the second formulation is an outer convexification, compare Section 2.1.

3.2. Results. We provide in [Table 1](#) a comparison of different solutions reported in the literature. The numbers show the respective lengths $t_i - t_{i-1}$ of the switching arcs with the value of $w(t)$ on the upper or lower bound (given in the second column). The infeasibility shows values obtained by a simulation with a Runge-Kutta-Fehlberg method of 4th/5th order and an integration tolerance of 10^{-8} .

TABLE 1

Results for the F-8 flight control problem. The solution in the second last column is a personal communication by Martin Schlüter and Matthias Gerdts.

Arc	$w(t)$	Lee[42]	Kaya[36]	Sager[53]	Schlüter	Sager
1	1	0.00000	0.10292	0.10235	0.0	1.13492
2	0	2.18800	1.92793	1.92812	0.608750	0.34703
3	1	0.16400	0.16687	0.16645	3.136514	1.60721
4	0	2.88100	2.74338	2.73071	0.654550	0.69169
5	1	0.33000	0.32992	0.32994	0.0	0.0
6	0	0.47200	0.47116	0.47107	0.0	0.0
Infeasibility		1.75E-3	1.64E-3	5.90E-6	3.29E-6	2.21E-7
Objective		6.03500	5.74218	5.72864	4.39981	3.78086

The best known optimal objective value of this problem given is given by $T = 3.78086$. The corresponding solution is shown in [Figure 1](#) (right), another local minimum is plotted in [Figure 1](#) (left). The solution of bang-bang type switches three resp. five times, starting with $w(t) = 1$.

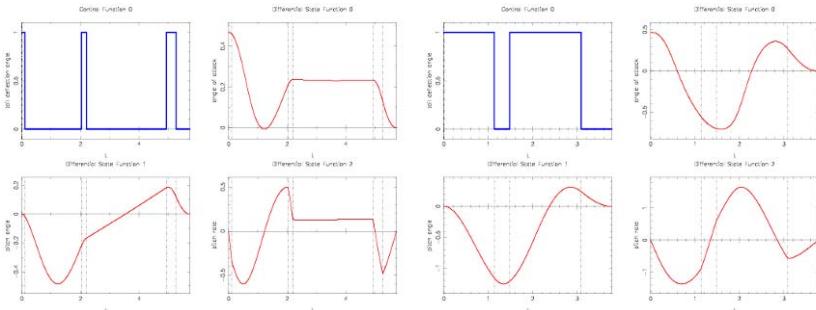


FIG. 1. Trajectories for the F-8 flight control problem. Left: corresponding to the Sager[53] column in Table 1. Right: corresponding to the rightmost column in Table 1.

4. Lotka Volterra fishing problem. The Lotka Volterra fishing problem seeks an optimal fishing strategy to be performed on a fixed time horizon to bring the biomasses of both predator as prey fish to a prescribed steady state. The problem was set up as a small-scale benchmark problem in [55] and has since been used for the evaluation of algorithms, e.g., [62].

The mathematical equations form a small-scale ODE model. The interior point equality conditions fix the initial values of the differential states. The optimal integer control shows chattering behavior, making the Lotka Volterra fishing problem an ideal candidate for benchmarking of algorithms.

4.1. Model and optimal control problem. The biomasses of two fish species — one predator, the other one prey — are the differential states of the model, the binary control is the operation of a fishing fleet. The optimization goal is to penalize deviations from a steady state,

$$\begin{aligned} \min_{x,w} \quad & \int_{t_0}^{t_f} (x_0 - 1)^2 + (x_1 - 1)^2 dt \\ \text{s.t.} \quad & \dot{x}_0 = x_0 - x_0 x_1 - c_0 x_0 w \\ & \dot{x}_1 = -x_1 + x_0 x_1 - c_1 x_1 w, \\ & x(0) = (0.5, 0.7)^T, \\ & w(t) \in \{0, 1\}, \quad t \in [0, t_f], \end{aligned} \quad (4.1)$$

with $t_f = 12$, $c_0 = 0.4$, and $c_1 = 0.2$.

4.2. Results. If the problem is relaxed, i.e., we demand that $w(\cdot)$ be in the continuous interval $[0, 1]$ instead of the binary choice $\{0, 1\}$, the optimal solution can be determined by means of Pontryagin's maximum principle [49]. The optimal solution contains a singular arc, [55].

The optimal objective value of this relaxed problem is $\Phi = 1.34408$. As follows from MIOC theory [58] this is the best lower bound on the optimal value of the original problem with the integer restriction on the control function. In other words, this objective value can be approximated arbitrarily close, if the control only switches often enough between 0 and 1. As no optimal solution exists, a suboptimal one is shown in Figure 2, with 26 switches and an objective function value of $\Phi = 1.34442$.

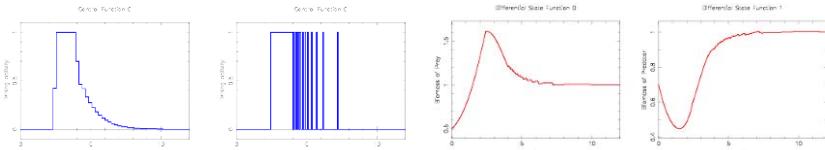


FIG. 2. Trajectories for the Lotka Volterra Fishing problem. Top left: optimal relaxed solution on grid with 52 intervals. Top right: feasible integer solution. Bottom: corresponding differential states, biomass of prey and of predator fish.

4.3. Variants. There are several alternative formulations and variants of the above problem, in particular

- a prescribed time grid for the control function [55],
- a time-optimal formulation to get into a steady-state [53],
- the usage of a different target steady-state, as the one corresponding to $w(\cdot) = 1$ which is $(1 + c_1, 1 - c_0)$,
- different fishing control functions for the two species,
- different parameters and start values.

5. Fuller’s problem. The first control problem with an optimal chattering solution was given by [23]. An optimal trajectory does exist for all initial and terminal values in a vicinity of the origin. As Fuller showed, this optimal trajectory contains a bang-bang control function that switches infinitely often. The mathematical equations form a small-scale ODE model. The interior point equality conditions fix initial and terminal values of the differential states, the objective is of tracking type.

5.1. Model and optimal control problem. The MIOCP reads as

$$\begin{aligned} \min_{x,w} \quad & \int_0^1 x_0^2 dt \\ \text{s.t.} \quad & \dot{x}_0 = x_1 \\ & \dot{x}_1 = 1 - 2w \\ & x(0) = (0.01, 0)^T, \quad x(T) = (0.01, 0)^T, \\ & w(t) \in \{0, 1\}, \quad t \in [0, 1]. \end{aligned} \tag{5.1}$$

5.2. Results. The optimal trajectories for the relaxed control problem on an equidistant grid \mathcal{G}^0 with $n_{\text{ms}} = 20, 30, 60$ are shown in the top row of Figure 3. Note that this solution is not bang-bang due to the discretization of the control space. Even if this discretization is made very fine, a trajectory with $w(\cdot) = 0.5$ on an interval in the middle of $[0, 1]$ will be found as a minimum.

The application of MS MINTOC [54] yields an objective value of $\Phi = 1.52845 \cdot 10^{-5}$, which is better than the limit of the relaxed problems, $\Phi^{20} = 1.53203 \cdot 10^{-5}$, $\Phi^{30} = 1.53086 \cdot 10^{-5}$, and $\Phi^{60} = 1.52958 \cdot 10^{-5}$.

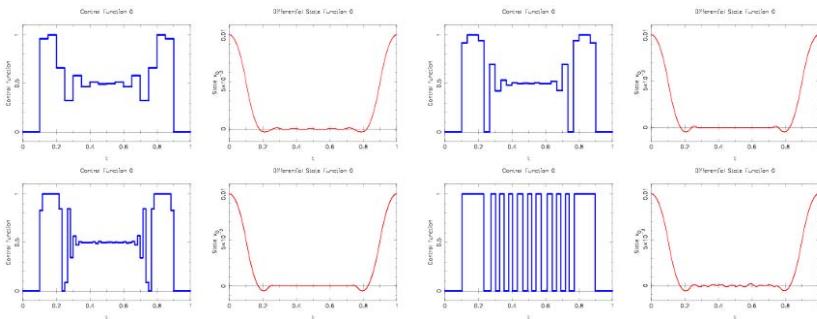


FIG. 3. Trajectories for Fuller’s problem. Top row and bottom left: relaxed optima for 20, 30, and 60 equidistant control intervals. Bottom right: feasible integer solution.

5.3. Variants. An extensive analytical investigation of this problem and a discussion of the ubiquity of Fuller’s problem can be found in [63].

6. Subway ride. The optimal control problem we treat in this section goes back to work of [9] for the city of New York. In an extension, also velocity limits that lead to path-constrained arcs appear. The aim is to minimize the energy used for a subway ride from one station to another, taking into account boundary conditions and a restriction on the time.

6.1. Model and optimal control problem. The MIOCP reads as

$$\begin{aligned} \min_{x,w} \quad & \int_0^{t_f} L(x,w) dt \\ \text{s.t.} \quad & \dot{x}_0 = x_1 \\ & \dot{x}_1 = f_1(x,w) \\ & x(0) = (0,0)^T, \quad x(t_f) = (2112,0)^T, \\ & w(t) \in \{1, 2, 3, 4\}, \quad t \in [0, t_f]. \end{aligned} \quad (6.1)$$

The terminal time $t_f = 65$ denotes the time of arrival of a subway train in the next station. The differential states $x_0(\cdot)$ and $x_1(\cdot)$ describe position and velocity of the train, respectively. The train can be operated in one of four different modes, $w(\cdot) = 1$ series, $w(\cdot) = 2$ parallel, $w(\cdot) = 3$ coasting, or $w(\cdot) = 4$ braking that accelerate or decelerate the train and have different energy consumption. Acceleration and energy comsumption are velocity-dependent. Hence, we will need switching functions $\sigma_i(x_1) = v_i - x_1$ for given velocities $v_i, i = 1..3$. The Lagrange term reads as

$$L(x, 1) = \begin{cases} e p_1 & \text{if } \sigma_1 \geq 0 \\ e p_2 & \text{else if } \sigma_2 \geq 0 \\ e \sum_{i=0}^5 c_i(1) \left(\frac{1}{10}\gamma x_1\right)^{-i} & \text{else} \end{cases} \quad (6.2)$$

$$L(x, 2) = \begin{cases} \infty & \text{if } \sigma_2 \geq 0 \\ e p_3 & \text{else if } \sigma_3 \geq 0 \\ e \sum_{i=0}^5 c_i(2) \left(\frac{1}{10}\gamma x_1 - 1\right)^{-i} & \text{else} \end{cases} \quad (6.3)$$

$$L(x, 3) = L(x, 4) = 0. \quad (6.4)$$

The right hand side function $f_1(x, w)$ reads as

$$f_1(x, 1) = \begin{cases} f_1^{1A} := \frac{g_e a_1}{W_{\text{eff}}} & \text{if } \sigma_1 \geq 0 \\ f_1^{1B} := \frac{g_e a_2}{W_{\text{eff}}} & \text{else if } \sigma_2 \geq 0 \\ f_1^{1C} := \frac{g_e T(x_1, 1) - R(x_1)}{W_{\text{eff}}} & \text{else} \end{cases} \quad (6.5)$$

$$f_1(x, 2) = \begin{cases} 0 & \text{if } \sigma_2 \geq 0 \\ f_1^{2B} := \frac{g_e a_3}{W_{\text{eff}}} & \text{else if } \sigma_3 \geq 0 \\ f_1^{2C} := \frac{g_e T(x_1, 2) - R(x_1)}{W_{\text{eff}}} & \text{else} \end{cases} \quad (6.6)$$

$$f_1(x, 3) = -\frac{g_e R(x_1)}{W_{\text{eff}}} - C, \quad (6.7)$$

$$f_1(x, 4) = -u = -u_{\max}. \quad (6.8)$$

TABLE 2
Parameters used for the subway MIOCP and its variants.

Symbol	Value	Unit	Symbol	Value	Unit
W	78000	lbs	v_1	0.979474	mph
W_{eff}	85200	lbs	v_2	6.73211	mph
S	2112	ft	v_3	14.2658	mph
S_4	700	ft	v_4	22.0	mph
S_5	1200	ft	v_5	24.0	mph
γ	$\frac{3600}{5280}$	$\frac{\text{sec}}{\text{h}} / \frac{\text{ft}}{\text{mile}}$	a_1	6017.611205	lbs
a	100	ft^2	a_2	12348.34865	lbs
n_{wag}	10	-	a_3	11124.63729	lbs
b	0.045	-	u_{\max}	4.4	ft / sec^2
C	0.367	-	p_1	106.1951102	-
g	32.2	$\frac{\text{ft}}{\text{sec}^2}$	p_2	180.9758408	-
e	1.0	-	p_3	354.136479	-

The braking deceleration $u(\cdot)$ can be varied between 0 and a given u_{\max} . It can be shown that for problem (6.1) only maximal braking can be optimal, hence we fixed $u(\cdot)$ to u_{\max} without loss of generality. Occurring forces are

$$R(x_1) = ca \gamma^2 x_1^2 + bW \gamma x_1 + \frac{1.3}{2000} W + 116, \quad (6.9)$$

$$T(x_1, 1) = \sum_{i=0}^5 b_i(1) \left(\frac{1}{10} \gamma x_1 - 0.3 \right)^{-i}, \quad (6.10)$$

$$T(x_1, 2) = \sum_{i=0}^5 b_i(2) \left(\frac{1}{10} \gamma x_1 - 1 \right)^{-i}. \quad (6.11)$$

Parameters are listed in [Table 2](#), while $b_i(w)$ and $c_i(w)$ are given by

$b_0(1)$	$-0.1983670410E02$,	$c_0(1)$	$0.3629738340E02$,
$b_1(1)$	$0.1952738055E03$,	$c_1(1)$	$-0.2115281047E03$,
$b_2(1)$	$0.2061789974E04$,	$c_2(1)$	$0.7488955419E03$,
$b_3(1)$	$-0.7684409308E03$,	$c_3(1)$	$-0.9511076467E03$,
$b_4(1)$	$0.2677869201E03$,	$c_4(1)$	$0.5710015123E03$,
$b_5(1)$	$-0.3159629687E02$,	$c_5(1)$	$-0.1221306465E03$,
$b_0(2)$	$-0.1577169936E03$,	$c_0(2)$	$0.4120568887E02$,
$b_1(2)$	$0.3389010339E04$,	$c_1(2)$	$0.3408049202E03$,
$b_2(2)$	$0.6202054610E04$,	$c_2(2)$	$-0.1436283271E03$,
$b_3(2)$	$-0.4608734450E04$,	$c_3(2)$	$0.8108316584E02$,
$b_4(2)$	$0.2207757061E04$,	$c_4(2)$	$-0.5689703073E01$,
$b_5(2)$	$-0.3673344160E03$,	$c_5(2)$	$-0.2191905731E01$.

Details about the derivation of this model and the assumptions made can be found in [9] or in [38].

6.2. Results. The optimal trajectory for this problem has been calculated by means of an indirect approach in [9, 38], and based on the direct multiple shooting method in [58]. The resulting trajectory is listed in [Table 3](#).

TABLE 3
Optimal trajectory for the subway MIOCP as calculated in [9, 38, 58].

Time t	$w(\cdot)$	$f_1 =$	x_0 [ft]	x_1 [mph]	x_1 [fps]	Energy
0.00000	1	f_1^{1A}	0.0	0.0	0.0	0.0
0.63166	1	f_1^{1B}	0.453711	0.979474	1.43656	0.0186331
2.43955	1	f_1^{1C}	10.6776	6.73211	9.87375	0.109518
3.64338	2	f_1^{2B}	24.4836	8.65723	12.6973	0.147387
5.59988	2	f_1^{2C}	57.3729	14.2658	20.9232	0.339851
12.6070	1	f_1^{1C}	277.711	25.6452	37.6129	0.93519
45.7827	3	$f_1(3)$	1556.5	26.8579	39.3915	1.14569
46.8938	3	$f_1(3)$	1600	26.5306	38.9115	1.14569
57.1600	4	$f_1(4)$	1976.78	23.5201	34.4961	1.14569
65.0000	-	-	2112	0.0	0.0	1.14569

6.3. Variants. The given parameters have to be modified to match different parts of the track, subway train types, or amount of passengers. A minimization of travel time might also be considered.

The problem becomes more challenging, when additional point or path constraints are considered. First we consider the point constraint

$$x_1 \leq v_4 \text{ if } x_0 = S_4 \quad (6.12)$$

for a given distance $0 < S_4 < S$ and velocity $v_4 > v_3$. Note that the state $x_0(\cdot)$ is strictly monotonically increasing with time, as $\dot{x}_0 = x_1 > 0$ for all $t \in (0, T)$.

The optimal order of gears for $S_4 = 1200$ and $v_4 = 22/\gamma$ with the additional interior point constraints (6.12) is 1, 2, 1, 3, 4, 2, 1, 3, 4. The stage lengths between switches are 2.86362, 10.722, 15.3108, 5.81821, 1.18383, 2.72451, 12.917, 5.47402, and 7.98594 with $\Phi = 1.3978$. For different parameters $S_4 = 700$ and $v_4 = 22/\gamma$ we obtain the gear choice 1, 2, 1, 3, 2, 1, 3, 4 and stage lengths 2.98084, 6.28428, 11.0714, 4.77575, 6.0483, 18.6081, 6.4893, and 8.74202 with $\Phi = 1.32518$.

A more practical restriction are path constraints on subsets of the track. We will consider a problem with additional path constraints

$$x_1 \leq v_5 \text{ if } x_0 \geq S_5. \quad (6.13)$$

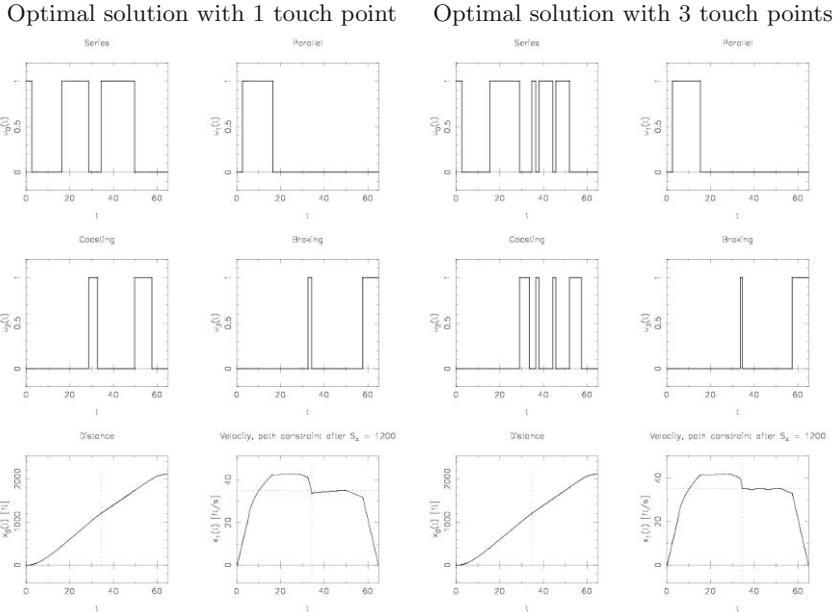


FIG. 4. The differential state velocity of a subway train over time. The dotted vertical line indicates the beginning of the path constraint, the horizontal line the maximum velocity. Left: one switch leading to one touch point. Right: optimal solution for three switches. The energy-optimal solution needs to stay as close as possible to the maximum velocity on this time interval to avoid even higher energy-intensive accelerations in the start-up phase to match the terminal time constraint $t_f \leq 65$ to reach the next station.

The additional path constraint changes the qualitative behavior of the relaxed solution. While all solutions considered this far were bang–bang and the main work consisted in finding the switching points, we now have a path-constraint arc. The optimal solutions for refined grids yield a series of monotonically decreasing objective function values, where the limit is the best value that can be approximated by an integer feasible solution. In our case we obtain

$$1.33108, 1.31070, 1.31058, 1.31058, \dots \quad (6.14)$$

Figure 4 shows two possible integer realizations, with a trade-off between energy consumption and number of switches. Note that the solutions approximate the optimal driving behavior (a convex combination of two operation modes) by switching between the two and causing a touching of the velocity constraint from below as many times as we switch.

7. Resetting calcium oscillations. The aim of the control problem is to identify strength and timing of inhibitor stimuli that lead to a phase singularity which annihilates intra-cellular calcium oscillations. This is formulated as an objective function that aims at minimizing the state

deviation from a desired unstable steady state, integrated over time. A calcium oscillator model describing intra-cellular calcium spiking in hepatocytes induced by an extracellular increase in adenosine triphosphate (ATP) concentration is described. The calcium signaling pathway is initiated via a receptor activated G-protein inducing the intra-cellular release of inositol triphosphate (IP₃) by phospholipase C. The IP₃ triggers the opening of endoplasmic reticulum and plasma membrane calcium channels and a subsequent inflow of calcium ions from intra-cellular and extracellular stores leading to transient calcium spikes.

The mathematical equations form a small-scale ODE model. The interior point equality conditions fix the initial values of the differential states. The problem is, despite of its low dimension, very hard to solve, as the target state is unstable.

7.1. Model and optimal control problem. The MIOCP reads as

$$\begin{aligned} \min_{x, w, w^{\max}} \quad & \int_0^{t_f} ||x(t) - \tilde{x}||_2^2 + p_1 w(t) \, dt \\ \text{s.t.} \quad & \dot{x}_0 = k_1 + k_2 x_0 - \frac{k_3 x_0 x_1}{x_0 + K_4} - \frac{k_5 x_0 x_2}{x_0 + K_6} \\ & \dot{x}_1 = k_7 x_0 - \frac{k_8 x_1}{x_1 + K_9} \\ & \dot{x}_2 = \frac{k_{10} x_1 x_2 x_3}{x_3 + K_{11}} + k_{12} x_1 + k_{13} x_0 - \frac{k_{14} x_2}{w \cdot x_2 + K_{15}} \\ & \quad - \frac{k_{16} x_2}{x_2 + K_{17}} + \frac{x_3}{10} \\ & \dot{x}_3 = -\frac{k_{10} x_1 x_2 x_3}{x_3 + K_{11}} + \frac{k_{16} x_2}{x_2 + K_{17}} - \frac{x_3}{10} \\ & x(0) = (0.03966, 1.09799, 0.00142, 1.65431)^T, \\ & 1.1 \leq w^{\max} \leq 1.3, \\ & w(t) \in \{1, w^{\max}\}, \quad t \in [0, t_f] \end{aligned} \quad (7.1)$$

with fixed parameter values $[t_0, t_f] = [0, 22]$, $k_1 = 0.09$, $k_2 = 2.30066$, $k_3 = 0.64$, $K_4 = 0.19$, $k_5 = 4.88$, $K_6 = 1.18$, $k_7 = 2.08$, $k_8 = 32.24$, $K_9 = 29.09$, $k_{10} = 5.0$, $K_{11} = 2.67$, $k_{12} = 0.7$, $k_{13} = 13.58$, $k_{14} = 153.0$, $K_{15} = 0.16$, $k_{16} = 4.85$, $K_{17} = 0.05$, $p_1 = 100$, and reference values $\tilde{x}_0 = 6.78677$, $\tilde{x}_1 = 22.65836$, $\tilde{x}_2 = 0.384306$, $\tilde{x}_3 = 0.28977$.

The differential states (x_0, x_1, x_2, x_3) describe concentrations of activated G-proteins, active phospholipase C, intra-cellular calcium, and intra-ER calcium, respectively. The external control $w(\cdot)$ is a temporally varying concentration of an uncompetitive inhibitor of the PMCA ion pump.

Modeling details can be found in [39]. In the given equations that stem from [41], the model is identical to the one derived there, except for an additional first-order leakage flow of calcium from the ER back to

the cytoplasm, which is modeled by $\frac{x_3}{10}$. It reproduces well experimental observations of cytoplasmic calcium oscillations as well as bursting behavior and in particular the frequency encoding of the triggering stimulus strength, which is a well known mechanism for signal processing in cell biology.

7.2. Results. The depicted optimal solution in Figure 5 consists of a stimulus of $w^{\max} = 1.3$ and a timing given by the stage lengths 4.6947115, 0.1491038, and 17.1561845. The optimal objective function value is $\Phi = 1610.654$. As can be seen from the additional plots, this solution is extremely unstable. A small perturbation in the control, or simply rounding errors on a longer time horizon lead to a transition back to the stable limit-cycle oscillations.

The determination of the stimulus by means of optimization is quite hard for two reasons. First, the unstable target steady-state. Only a stable all-at-once algorithm such as multiple shooting or collocation can be applied successfully. Second, the objective landscape of the problem in switching time formulation (this is, for a fixed stimulus strength and modifying only beginning and length of the stimulus) is quite nasty, as the visualizations in [53] and on the web page [52] show.

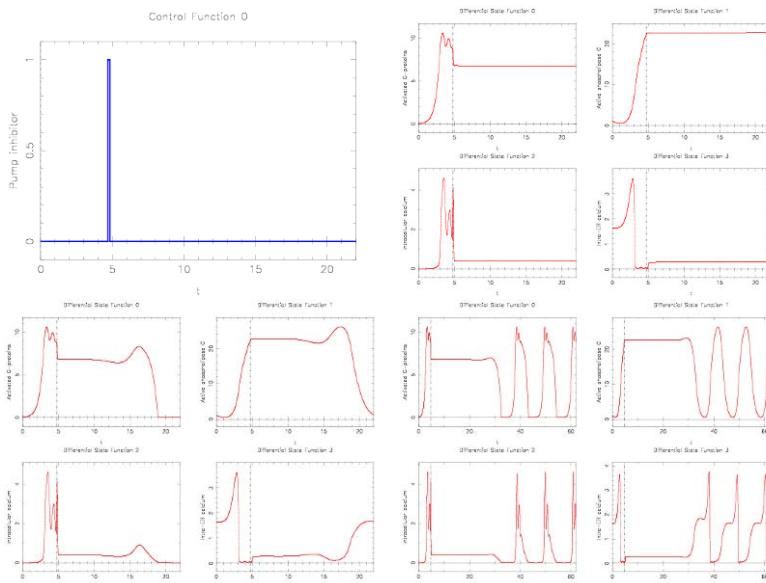


FIG. 5. Trajectories for the calcium problem. Top left: optimal integer solution. Top right: corresponding differential states with phase resetting. Bottom left: slightly perturbed control: stimulus 0.001 too early. Bottom right: long time behavior of optimal solution: numerical rounding errors lead to transition back from unstable steady-state to stable limit-cycle.

7.3. Variants. Alternatively, also the annihilation of calcium oscillations with PLC activation inhibition, i.e., the use of two control functions is possible, compare [41]. Of course, results depend very much on the scaling of the deviation in the objective function.

8. Supermarket refrigeration system. This benchmark problem was formulated first within the European network of excellence HYCON, [19] by Larsen et. al, [40]. The formulation lacks however a precise definition of initial values and constraints, which are only formulated as “soft constraints”. The task is to control a refrigeration system in an energy optimal way, while guaranteeing safeguards on the temperature of the showcases. This problem would typically be a moving horizon online optimization problem, here it is defined as a fixed horizon optimization task.

The mathematical equations form a periodic ODE model.

8.1. Model and optimal control problem. The MIOCP reads as

$$\begin{aligned} \min_{x,w,t_f} \quad & \frac{1}{t_f} \int_0^{t_f} (w_2 + w_3) \cdot 0.5 \cdot \eta_{vol} \cdot V_{sl} \cdot f dt \\ \text{s.t.} \quad & \dot{x}_0 = \frac{\left(x_4(x_2 - T_e(x_0)) + x_8(x_6 - T_e(x_0)) \right)}{V_{suc} \cdot \frac{d\rho_{suc}}{dP_{suc}}(x_0)} \cdot \frac{UA_{wrm}}{M_{rm} \cdot \Delta h_{lg}(x_0)} \\ & + \frac{M_{rc} - \eta_{vol} \cdot V_{sl} \cdot 0.5 (w_2 + w_3) \rho_{suc}(x_0)}{V_{suc} \cdot \frac{d\rho_{suc}}{dP_{suc}}(x_0)} \\ & \dot{x}_1 = -\frac{UA_{goods-air}(x_1 - x_3)}{M_{goods} \cdot C_{p,goods}} \\ & \dot{x}_2 = \frac{UA_{air-wall}(x_3 - x_2) - \frac{UA_{wrm}}{M_{rm}} x_4(x_2 - T_e(x_0))}{M_{wall} \cdot C_{p,wall}} \\ & \dot{x}_3 = \frac{UA_{goods-air}(x_1 - x_3) + \dot{Q}_{airload} - UA_{air-wall}(x_3 - x_2)}{M_{air} \cdot C_{p,air}} \\ & \dot{x}_4 = \left(\frac{M_{rm} - x_4}{\tau_{fill}} \right) w_0 - \frac{UA_{wrm}(1 - w_0)}{M_{rm} \cdot \Delta h_{lg}(x_0)} x_4(x_2 - T_e(x_0)) \\ & \dot{x}_5 = -\frac{UA_{goods-air}(x_5 - x_7)}{M_{goods} \cdot C_{p,goods}} \\ & \dot{x}_6 = \frac{UA_{air-wall}(x_7 - x_6) - \frac{UA_{wrm}}{M_{rm}} x_8(x_6 - T_e(x_0))}{M_{wall} \cdot C_{p,wall}} \\ & \dot{x}_7 = \frac{UA_{goods-air}(x_5 - x_7) + \dot{Q}_{airload} - UA_{air-wall}(x_7 - x_6)}{M_{air} \cdot C_{p,air}} \end{aligned}$$

TABLE 4
Parameters used for the supermarket refrigeration problem.

Symbol	Value	Unit	Description
$\dot{Q}_{airload}$	3000.00	$\frac{J}{s}$	Disturbance, heat transfer
\dot{m}_{rc}	0.20	$\frac{kg}{s}$	Disturbance, constant mass flow
M_{goods}	200.00	kg	Mass of goods
$C_{p,goods}$	1000.00	$\frac{J}{kg \cdot K}$	Heat capacity of goods
$UA_{goods-air}$	300.00	$\frac{J}{s \cdot K}$	Heat transfer coefficient
M_{wall}	260.00	kg	Mass of evaporator wall
$C_{p,wall}$	385.00	$\frac{J}{kg \cdot K}$	Heat capacity of evaporator wall
$UA_{air-wall}$	500.00	$\frac{J}{s \cdot K}$	Heat transfer coefficient
M_{air}	50.00	kg	Mass of air in display case
$C_{p,air}$	1000.00	$\frac{J}{kg \cdot K}$	Heat capacity of air
UA_{wrm}	4000.00	$\frac{J}{s \cdot K}$	Maximum heat transfer coefficient
τ_{fill}	40.00	s	Filling time of the evaporator
T_{SH}	10.00	K	Superheat in the suction manifold
M_{rm}	1.00	kg	Maximum mass of refrigerant
V_{suc}	5.00	m^3	Total volume of suction manifold
V_{sl}	0.08	$\frac{m^3}{s}$	Total displacement volume
η_{vol}	0.81	—	Volumetric efficiency

$$\begin{aligned} \dot{x}_8 &= \left(\frac{M_{rm} - x_8}{\tau_{fill}} \right) w_1 - \frac{UA_{wrm}(1 - w_1)}{M_{rm} \cdot \Delta h_{lg}(x_0)} x_8(x_6 - T_e(x_0)) \\ x(0) &= x(t_f), \\ 650 \leq t_f &\leq 750, \\ x_0 &\leq 1.7, \quad 2 \leq x_3 \leq 5, \quad 2 \leq x_7 \leq 5 \\ w(t) &\in \{0, 1\}^4, \quad t \in [0, t_f]. \end{aligned}$$

The differential state x_0 describes the suction pressure in the suction manifold (in bar). The next three states model temperatures in the first display case (in C). x_1 is the goods' temperature, x_2 the one of the evaporator wall and x_3 the air temperature surrounding the goods. x_4 then models the mass of the liquefied refrigerant in the evaporator (in kg). x_5 to x_8 describe the corresponding states in the second display case. w_0 and w_1 describe the inlet valves of the first two display cases, respectively. w_2 and w_3 denote the activity of a single compressor.

The model uses the parameter values listed in Table 4 and the polynomial functions obtained from interpolations:

$$\begin{aligned}
T_e(x_0) &= -4.3544x_0^2 + 29.224x_0 - 51.2005, \\
\Delta h_{lg}(x_0) &= (0.0217x_0^2 - 0.1704x_0 + 2.2988) \cdot 10^5, \\
\rho_{suc}(x_0) &= 4.6073x_0 + 0.3798, \\
\frac{dp_{suc}}{dP_{suc}}(x_0) &= -0.0329x_0^3 + 0.2161x_0^2 - 0.4742x_0 + 5.4817.
\end{aligned}$$

8.2. Results. For the relaxed problem the optimal solution is $\Phi = 12072.45$. The integer solution plotted in Figure 6 is feasible, but yields an increased objective function value of $\Phi = 12252.81$, a compromise between effectiveness and a reduced number of switches.

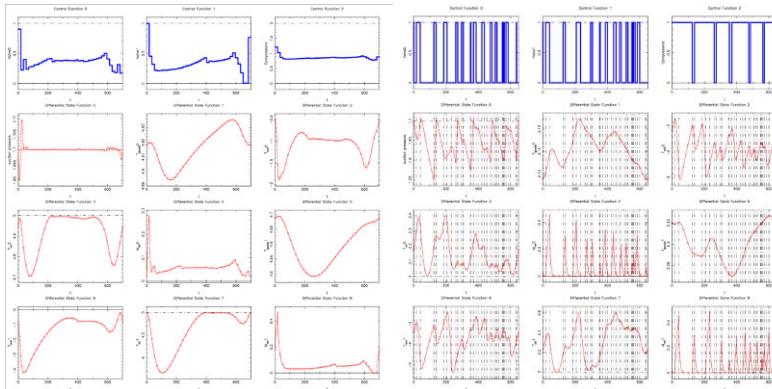


FIG. 6. Periodic trajectories for optimal relaxed (left) and integer feasible controls (right), with the controls $w(\cdot)$ in the first row and the differential states in the three bottom rows.

8.3. Variants. Since the compressors are parallel connected one can introduce a single control $w_2 \in \{0, 1, 2\}$ instead of two equivalent controls. The same holds for scenarios with n parallel connected compressors.

In [40], the problem was stated slightly different:

- The temperature constraints weren't hard bounds but there was a penalization term added to the objective function to minimize the violation of these constraints.
- The differential equation for the mass of the refrigerant had another switch, if the valve (e.g. w_0) is closed. It was formulated as $\dot{x}_4 = \frac{M_{rm} - x_4}{\tau_{fill}}$ if $w_0 = 1$, $\dot{x}_4 = -\frac{UA_{wrm}}{M_{rm} \cdot \Delta h_{lg}(x_0)} x_4 (x_2 - T_e(x_0))$ if $w_0 = 0$ and $x_4 > 0$, or $\dot{x}_4 = 0$ if $w_0 = 0$ and $x_4 = 0$. This additional switch is redundant because the mass itself is a factor on the right hand side and so the complete right hand side is 0 if $x_4 = 0$.
- A night scenario with two different parameters was given. At night the following parameters change their value to $\dot{Q}_{airload} = 1800.00 \frac{J}{s}$

and $\dot{m}_{rc} = 0.00 \frac{kg}{s}$. Additionally the constraint on the suction pressure $x_0(t)$ is softened to $x_0(t) \leq 1.9$.

- The number of compressors and display cases is not fixed. Larsen also proposed the problem with 3 compressors and 3 display cases. This leads to a change in the compressor rack's performance to $V_{sl} = 0.095 \frac{m^3}{s}$. Unfortunately this constant is only given for these two cases although Larsen proposed scenarios with more compressors and display cases.

9. Elchtest testdrive. We consider a time-optimal car driving maneuver to avoid an obstacle with small steering effort. At any time, the car must be positioned on a prescribed track. This control problem was first formulated in [25] and used for subsequent studies [26, 37].

The mathematical equations form a small-scale ODE model. The interior point equality conditions fix initial and terminal values of the differential states, the objective is of minimum-time type.

9.1. Model and optimal control problem. We consider a car model derived under the simplifying assumption that rolling and pitching of the car body can be neglected. Only a single front and rear wheel is modelled, located in the virtual center of the original two wheels. Motion of the car body is considered on the horizontal plane only.

The MIOCP reads as

$$\min_{t_f, x(\cdot), u(\cdot)} t_f + \int_0^{t_f} w_\delta^2(t) dt \quad (9.1a)$$

$$\text{s.t.} \quad \dot{c}_x = v \cos(\psi - \beta) \quad (9.1b)$$

$$\dot{c}_y = v \sin(\psi - \beta) \quad (9.1c)$$

$$\begin{aligned} \dot{v} = \frac{1}{m} & \left((F_{lr}^\mu - F_{Ax}) \cos \beta + F_{lf} \cos(\delta + \beta) \right. \\ & \left. - (F_{sr} - F_{Ay}) \sin \beta - F_{sf} \sin(\delta + \beta) \right) \end{aligned} \quad (9.1d)$$

$$\dot{\delta} = w_\delta \quad (9.1e)$$

$$\begin{aligned} \dot{\beta} = w_z - \frac{1}{m v} & \left((F_{lr} - F_{Ax}) \sin \beta + F_{lf} \sin(\delta + \beta) \right. \\ & \left. + (F_{sr} - F_{Ay}) \cos \beta + F_{sf} \cos(\delta + \beta) \right) \end{aligned} \quad (9.1f)$$

$$\dot{\psi} = w_z \quad (9.1g)$$

$$\dot{w}_z = \frac{1}{I_{zz}} (F_{sf} l_f \cos \delta - F_{sr} l_r - F_{Ay} e_{SP} + F_{lf} l_f \sin \delta) \quad (9.1h)$$

$$c_y(t) \in [P_l(c_x(t)) + \frac{B}{2}, P_u(c_x(t)) - \frac{B}{2}] \quad (9.1i)$$

$$w_\delta(t) \in [-0.5, 0.5], \quad F_B(t) \in [0, 1.5 \cdot 10^4], \quad \phi(t) \in [0, 1] \quad (9.1j)$$

$$\mu(t) \in \{1, \dots, 5\} \quad (9.1k)$$

$$x(t_0) = (-30, \text{free}, 10, 0, 0, 0, 0, 0)^\top, \quad (c_x, \psi)(t_f) = (140, 0) \quad (9.1l)$$

for $t \in [t_0, t_f]$ almost everywhere. The four control functions contained in $u(\cdot)$ are steering wheel angular velocity w_δ , total braking force F_B , the accelerator pedal position ϕ and the gear μ . The differential states contained in $x(\cdot)$ are horizontal position of the car c_x , vertical position of the car c_y , magnitude of directional velocity of the car v , steering wheel angle δ , side slip angle β , yaw angle ψ , and the yaw angle velocity w_z .

The model parameters are listed in [Table 5](#), while the forces and expressions in (9.1b) to (9.1h) are given for fixed μ by

$$\begin{aligned}
F_{sf,sr}(\alpha_{f,r}) &:= D_{f,r} \sin \left(C_{f,r} \arctan(B_{f,r} \alpha_{f,r} \right. \\
&\quad \left. - E_{f,r}(B_{f,r} \alpha_{f,r} - \arctan(B_{f,r} \alpha_{f,r})) \right), \\
\alpha_f &:= \delta(t) - \arctan \left(\frac{l_f \dot{\psi}(t) - v(t) \sin \beta(t)}{v(t) \cos \beta(t)} \right) \\
\alpha_r &:= \arctan \left(\frac{l_r \dot{\psi}(t) + v(t) \sin \beta(t)}{v(t) \cos \beta(t)} \right), \\
F_{lf} &:= -F_{Bf} - F_{Rf}, \\
F_{lr}^\mu &:= \frac{i_g^\mu i_t}{R} M_{mot}^\mu(\phi) - F_{Br} - F_{Rr}, \\
M_{mot}^\mu(\phi) &:= f_1(\phi) f_2(w_{mot}^\mu) + (1 - f_1(\phi)) f_3(w_{mot}^\mu), \\
f_1(\phi) &:= 1 - \exp(-3 \phi), \\
f_2(w_{mot}) &:= -37.8 + 1.54 w_{mot} - 0.0019 w_{mot}^2, \\
f_3(w_{mot}) &:= -34.9 - 0.04775 w_{mot}, \\
w_{mot}^\mu &:= \frac{i_g^\mu i_t}{R} v(t), \\
F_{Bf} &:= \frac{2}{3} F_B, \quad F_{Br} := \frac{1}{3} F_B, \\
F_{Rf}(v) &:= f_R(v) \frac{m l_r g}{l_f + l_r}, \quad F_{Rr}(v) := f_R(v) \frac{m l_f g}{l_f + l_r}, \\
f_R(v) &:= 9 \cdot 10^{-3} + 7.2 \cdot 10^{-5} v + 5.038848 \cdot 10^{-10} v^4, \\
F_{Ax} &:= \frac{1}{2} c_w \rho A v^2(t), \quad F_{Ay} := 0.
\end{aligned}$$

The test track is described by setting up piecewise cubic spline functions $P_l(x)$ and $P_r(x)$ modeling the top and bottom track boundary, given a horizontal position x .

TABLE 5
Parameters used in the car model.

	Value	Unit	Description
m	$1.239 \cdot 10^3$	kg	Mass of the car
g	9.81	$\frac{\text{m}}{\text{s}^2}$	Gravity constant
l_f	1.19016	m	Front wheel distance to c.o.g.
l_r	1.37484	m	Rear wheel distance to c.o.g.
R	0.302	m	Wheel radius
I_{zz}	$1.752 \cdot 10^3$	kg m^2	Moment of inertia
c_w	0.3	—	Air drag coefficient
ρ	1.249512	$\frac{\text{kg}}{\text{m}^3}$	Air density
A	1.4378946874	m^2	Effective flow surface
i_g^1	3.09	—	Gear 1 transmission ratio
i_g^2	2.002	—	Gear 2 transmission ratio
i_g^3	1.33	—	Gear 3 transmission ratio
i_g^4	1.0	—	Gear 4 transmission ratio
i_g^5	0.805	—	Gear 5 transmission ratio
i_t	3.91	—	Engine torque transmission
B_f	$1.096 \cdot 10^1$	—	Pacejka coeff. (stiffness)
B_r	$1.267 \cdot 10^1$	—	
$C_{f,r}$	1.3	—	Pacejka coefficients (shape)
D_f	$4.5604 \cdot 10^3$	—	Pacejka coefficients (peak)
D_r	$3.94781 \cdot 10^3$	—	
$E_{f,r}$	-0.5	—	Pacejka coefficients (curv.)

$$P_l(x) := \begin{cases} 0 & \text{if } x \leq 44, \\ 4 h_2 (x - 44)^3 & \text{if } 44 < x \leq 44.5, \\ 4 h_2 (x - 45)^3 + h_2 & \text{if } 44.5 < x \leq 45, \\ h_2 & \text{if } 45 < x \leq 70, \\ 4 h_2 (70 - x)^3 + h_2 & \text{if } 70 < x \leq 70.5, \\ 4 h_2 (71 - x)^3 & \text{if } 70.5 < x \leq 71, \\ 0 & \text{if } 71 < x. \end{cases} \quad (9.2)$$

$$P_u(x) := \begin{cases} h_1 & \text{if } x \leq 15, \\ 4 (h_3 - h_1) (x - 15)^3 + h_1 & \text{if } 15 < x \leq 15.5, \\ 4 (h_3 - h_1) (x - 16)^3 + h_3 & \text{if } 15.5 < x \leq 16, \\ h_3 & \text{if } 16 < x \leq 94, \\ 4 (h_3 - h_4) (94 - x)^3 + h_3 & \text{if } 94 < x \leq 94.5, \\ 4 (h_3 - h_4) (95 - x)^3 + h_4 & \text{if } 94.5 < x \leq 95, \\ h_4 & \text{if } 95 < x. \end{cases} \quad (9.3)$$

where $B = 1.5$ m is the car's width and

$$h_1 := 1.1 B + 0.25, \quad h_2 := 3.5, \quad h_3 := 1.2 B + 3.75, \quad h_4 := 1.3 B + 0.25.$$

9.2. Results. In [25, 26, 37] numerical results for the benchmark problem have been deduced. In [37] one can also find an explanation why a bang-bang solution for the relaxed and convexified gear choices has to be optimal. Table 6 gives the optimal gear choice and the resulting objective function value (the end time) for different numbers N of control discretization intervals, which were also used for a discretization of the path constraints.

TABLE 6

Gear choice depending on discretization in time N . Times when gear becomes active.

N	$\mu = 1$	$\mu = 2$	$\mu = 3$	$\mu = 4$	$\mu = 5$	t_f
10	0.0	0.435956	2.733326	—	—	6.764174
20	0.0	0.435903	2.657446	6.467723	—	6.772046
40	0.0	0.436108	2.586225	6.684504	—	6.782052
80	0.0	0.435796	2.748930	6.658175	—	6.787284

10. Elliptic track testdrive. This control problem is very similar to the one in Section 9. However, instead of a simple lane change maneuver the time-optimal driving on an elliptic track with periodic boundary conditions is considered, [57].

10.1. Model and optimal control problem. With the notation of Section 9 the MIOCP reads as

$$\begin{aligned} & \min_{t_f, x(\cdot), u(\cdot)} && t_f \\ & \text{s.t.} && (9.1b - 9.1h), (9.1j), (9.1k), \\ & && (c_x, c_y) \in \mathcal{X}, \\ & && x(t_0) = x(t_f) - (0, 0, 0, 0, 0, 2\pi, 0)^T, \\ & && c_y(t_0) = 0, \\ & && 0 \leq r^{\text{eng}}(v, \mu), \end{aligned} \tag{10.1a}$$

for $t \in [t_0, t_f]$ almost everywhere.

The set \mathcal{X} describes an elliptic track with axes of $a = 170$ meters and $b = 80$ meters respectively, centered in the origin. The track's width is $W = 7.5$ meters, five times the car's width $B = 1.5$ meters,

$$\mathcal{X} = \left\{ [(a + r) \cos \eta, (b + r) \sin \eta] \mid r \in [-W/2, W/2] \subset \mathbb{R} \right\},$$

with $\eta = \arctan \frac{c_y}{c_x}$. Note that the special case $c_x = 0$ leading to $\eta = \pm \frac{\pi}{2}$ requires separate handling.

The model in Section 9 has a shortcoming, as switching to a low gear is possible also at high velocities, although this would lead to an unphysically

high engine speed. Therefore we extend it by additional constraints on the car's engine speed

$$800 =: n_{\text{eng}}^{\text{MIN}} \leq n_{\text{eng}} \leq n_{\text{eng}}^{\text{MAX}} := 8000, \quad (10.2)$$

in the form of equivalent velocity constraints

$$\frac{\pi n_{\text{eng}}^{\text{MIN}} R}{30 i_t i_g^\mu} \leq v \leq \frac{\pi n_{\text{eng}}^{\text{MAX}} R}{30 i_t i_g^\mu} \quad (10.3)$$

for all $t \in [0, t_f]$ and the active gear μ . We write this as $r^{\text{eng}}(v, \mu) \geq 0$.

10.2. Results. Parts of the optimal trajectory from [57] are shown in Figures 7 and 8. The order of gears is $(2, 3, 4, 3, 2, 1, 2, 3, 4, 3, 2, 1, 2)$. The gear switches take place after 1.87, 5.96, 10.11, 11.59, 12.21, 12.88, 15.82, 19.84, 23.99, 24.96, 26.10, and 26.76 seconds, respectively. The final time is $t_f = 27.7372$ s.

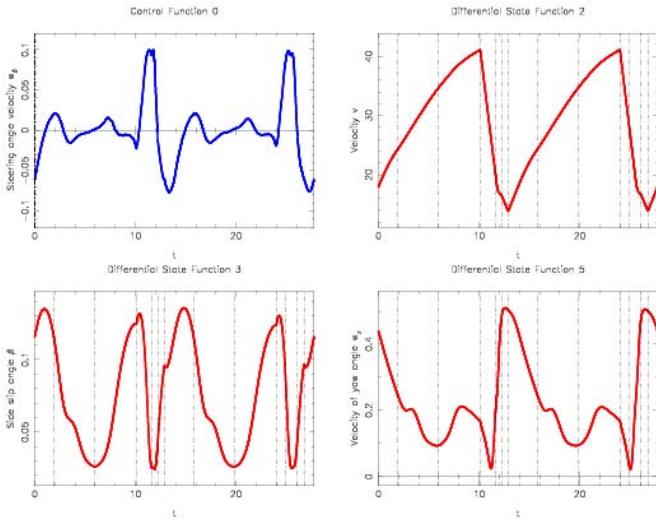


FIG. 7. The steering angle velocity (control), and some differential states of the optimal solution: directional velocity, side slip angle β , and velocity of yaw angle w_z plotted over time. The vertical lines indicate gear shifts.

As can be seen in Fig. 8, the car uses the track width to its full extent, leading to active path constraints. As was expected, the optimal gear increases in an acceleration phase. When the velocity has to be reduced, a combination of braking, no acceleration, and engine brake is used.

The result depends on the engine speed constraint $r^{\text{eng}}(v, \mu)$ that becomes active in the braking phase. If the constraint is omitted, the optimal

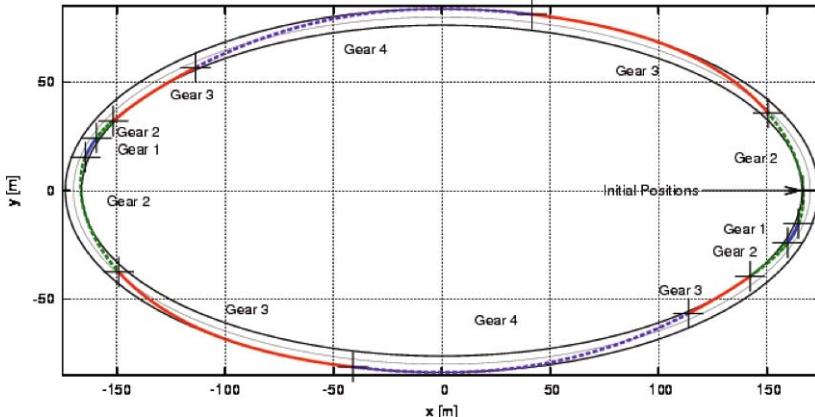


FIG. 8. Elliptic race track seen from above with optimal position and gear choices of the car. Note the exploitation of the slip (sliding) to change the car's orientation as fast as possible, when in first gear. The gear order changes when a different maximum engine speed is imposed.

solution switches directly from the fourth gear into the first one to maximize the effect of the engine brake. For $n_{\text{eng}}^{\text{MAX}} = 15000$ braking occurs in the gear order 4, 2, 1.

Although this was left as a degree of freedom, the optimizer yields a symmetric solution with respect to the upper and lower parts of the track for all scenarios we considered.

10.3. Variants. By a more flexible use of Bezier patches more general track constraints can be specified, e.g., of formula 1 race courses.

11. Simulated moving bed. We consider a simplified model of a Simulated Moving Bed (SMB) chromatographic separation process that contains time-dependent discrete decisions. SMB processes have been gaining increased attention lately, see [17, 34, 56] for further references. The related optimization problems are challenging from a mathematical point of view, as they combine periodic nonlinear optimal control problems in partial differential equations (PDE) with time-dependent discrete decisions.

11.1. Model and optimal control problem. SMB chromatography finds various industrial applications such as sugar, food, petrochemical and pharmaceutical industries. A SMB unit consists of multiple columns filled with solid absorbent. The columns are connected in a continuous cycle. There are two inlet streams, *desorbent* (De) and *feed* (Fe), and two outlet streams, *raffinate* (Ra) and *extract* (Ex). The continuous counter-current operation is simulated by switching the four streams periodically in the direction of the liquid flow in the columns, thereby leading to better separation. This is visualized in Figure 9.

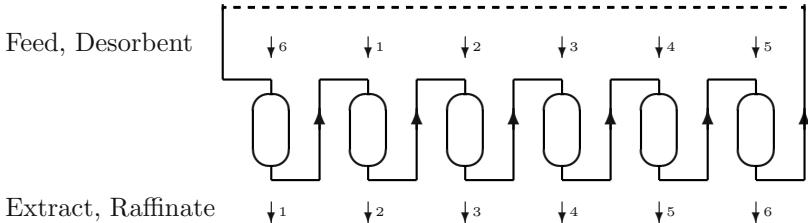


FIG. 9. Scheme of SMB process with 6 columns.

Due to this discrete switching of columns, SMB processes reach a cyclic or periodic steady state, i.e., the concentration profiles at the end of a period are equal to those at the beginning shifted by one column ahead in direction of the fluid flow. A number of different operating schemes have been proposed to further improve the performance of SMB.

The considered SMB unit consists of $N_{\text{col}} = 6$ columns. The flow rate through column i is denoted by Q_i , $i \in I := \{1, \dots, N_{\text{col}}\}$. The raffinate, desorbent, extract and feed flow rates are denoted by Q_{Ra} , Q_{De} , Q_{Ex} and Q_{Fe} , respectively. The (possibly) time-dependent value $w_{i\alpha}(t) \in \{0, 1\}$ denotes if the port of flow $\alpha \in \{\text{Ra}, \text{De}, \text{Ex}, \text{Fe}\}$ is positioned at column $i \in I$. As in many practical realizations of SMB processes only one pump per flow is available and the ports are switched by a 0–1 valve, we obtain the additional *special ordered set type one* restriction

$$\sum_{i \in I} w_{i\alpha}(t) = 1, \quad \forall t \in [0, T], \alpha \in \{\text{Ra}, \text{De}, \text{Ex}, \text{Fe}\}. \quad (11.1)$$

The flow rates Q_1 , Q_{De} , Q_{Ex} and Q_{Fe} enter as control functions $u(\cdot)$ resp. time-invariant parameters p into the optimization problem, depending on the operating scheme to be optimized. The remaining flow rates are derived by mass balance as

$$Q_{\text{Ra}} = Q_{\text{De}} - Q_{\text{Ex}} + Q_{\text{Fe}} \quad (11.2)$$

$$Q_i = Q_{i-1} - \sum_{\alpha \in \{\text{Ra}, \text{Ex}\}} w_{i\alpha} Q_\alpha + \sum_{\alpha \in \{\text{De}, \text{Fe}\}} w_{i\alpha} Q_\alpha \quad (11.3)$$

for $i = 2, \dots, N_{\text{col}}$. The feed contains two components A and B dissolved in desorbent, with concentrations $c_{\text{Fe}}^A = c_{\text{Fe}}^B = 0.1$. The concentrations of A and B in desorbent are $c_{\text{De}}^A = c_{\text{De}}^B = 0$.

A simplified equilibrium model is described in Diehl and Walther [16]. It can be derived from an equilibrium assumption between solid and liquid phases along with a simple spatial discretization. The mass balance in the liquid phase for $K = \text{A}, \text{B}$ is given by:

$$\epsilon_b \frac{\partial c_i^K(x, t)}{\partial t} + (1 - \epsilon_b) \frac{\partial q_i^K(x, t)}{\partial t} + u_i(t) \frac{\partial c_i^K(x, t)}{\partial x} = 0 \quad (11.4)$$

with equilibrium between the liquid and solid phases given by a linear isotherm:

$$q_i^K(x, t) = C_K c_i^K(x, t). \quad (11.5)$$

Here ϵ_b is the void fraction, $c_i^K(x, t)$ is the concentration in the liquid phase of component K in column i , q_i^K is the concentration in the solid phase. Also, i is the column index and N_{Column} is the number of columns. We can combine (11.4) and (11.5) and rewrite the model as:

$$\frac{\partial c_i^K(x, t)}{\partial t} = -(u_i(t)/\bar{K}_K) \frac{\partial c_i^K(x, t)}{\partial x} \quad (11.6)$$

where $\bar{K}_K = \epsilon_b + (1 - \epsilon_b)C_K$. Dividing the column into N_{FEX} compartments and applying a simple backward difference with $\Delta x = L/N_{FEX}$ leads to:

$$\frac{dc_{i,j}^K}{dt} = \frac{u_i(t)N_{FEX}}{\bar{K}_K L} [c_{i,j-1}^K(t) - c_{i,j}^K(t)] = k^K [c_{i,j-1}^K(t) - c_{i,j}^K(t)] \quad (11.7)$$

for $j = 1, \dots, N_{FEX}$, with $k^A = 2N_{FEX}$, $k^B = N_{FEX}$, and $c_{i,j}^K(t)$ is a discretization of $c_i^K(j\Delta x, t)$ for $j = 0, \dots, N_{FEX}$.

This simplified model for the dynamics in each column considers axial convection and axial mixing introduced by dividing the respective column into N_{dis} perfectly mixed compartments. Although this simple discretization does not consider all effects present in the advection–diffusion equation for the time and space dependent concentrations, the qualitative behavior of the concentration profiles moving at different velocities through the respective columns is sufficiently well represented. We assume that the compartment concentrations are constant. We denote the concentrations of A and B in the compartment with index i by c_i^A , c_i^B and leave away the time dependency. For the first compartment $j = (i-1)N_{\text{dis}} + 1$ of column $i \in I$ we have by mass transfer for $K = A, B$

$$\frac{\dot{c}_j^K}{k^K} = Q_i^- c_{j^-}^K - Q_i c_j^K - \sum_{\alpha \in \{\text{Ra,Ex}\}} w_{i\alpha} Q_\alpha c_{j^-}^K + \sum_{\alpha \in \{\text{De,Fe}\}} w_{i\alpha} Q_\alpha C_\alpha^K \quad (11.8)$$

where i^- is the preceding column, $i^- = N_{\text{col}}$ if $i = 1$, $i^- = i - 1$, else and equivalently $j^- = N$ if $j = 1$, $j^- = j - 1$, else. k^K denotes the axial convection in the column, $k^A = 2N_{\text{dis}}$ and $k^B = N_{\text{dis}}$. Component A is less adsorbed, thus travels faster and is prevailing in the raffinate, while B travels slower and is prevailing in the extract. For interior compartments j in column i we have

$$\frac{\dot{c}_j^K}{k^K} = Q_{i^-} c_{j^-}^K - Q_i c_j^K. \quad (11.9)$$

The compositions of extract and raffinate, $\alpha \in \{\text{Ex}, \text{Ra}\}$, are given by

$$\dot{M}_\alpha^K = Q_\alpha \sum_{i \in I} w_{i\alpha} c_{j(i)}^K \quad (11.10)$$

with $j(i)$ the last compartment of column i^- . The feed consumption is

$$\dot{M}_{\text{Fe}} = Q_{\text{Fe}}. \quad (11.11)$$

These are altogether $2N + 5$ differential equations for the differential states $x = (x_A, x_B, x_M)$ with $x_A = (c_0^A, \dots, c_N^A)$, $x_B = (c_0^B, \dots, c_N^B)$, and finally $x_M = (M_{\text{Ex}}^A, M_{\text{Ex}}^B, M_{\text{Ra}}^A, M_{\text{Ra}}^B, M_{\text{Fe}})$. They can be summarized as

$$\dot{x}(t) = f(x(t), u(t), w(t), p). \quad (11.12)$$

We define a linear operator $P : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x}$ that shifts the concentration profiles by one column and sets the auxiliary states to zero, i.e.,

$$\begin{aligned} x &\mapsto Px := (P_A x_A, P_B x_B, P_M x_M) \quad \text{with} \\ P_A x_A &:= (c_{N_{\text{dis}}+1}^A, \dots, c_N^A, c_1^A, \dots, c_{N_{\text{dis}}}^A), \\ P_B x_B &:= (c_{N_{\text{dis}}+1}^B, \dots, c_N^B, c_1^B, \dots, c_{N_{\text{dis}}}^B), \\ P_M x_M &:= (0, 0, 0, 0, 0). \end{aligned}$$

Then we can impose periodicity after the unknown cycle duration T by requiring $x(0) = Px(T)$. The purity of component A in the raffinate at the end of the cycle must be higher than $p_{\text{Ra}} = 0.95$ and the purity of B in the extract must be higher than $p_{\text{Ex}} = 0.95$, i.e., we impose the terminal purity conditions

$$M_{\text{Ex}}^A(T) \leq \frac{1 - p_{\text{Ex}}}{p_{\text{Ex}}} M_{\text{Ex}}^B(T), \quad (11.13)$$

$$M_{\text{Ra}}^B(T) \leq \frac{1 - p_{\text{Ra}}}{p_{\text{Ra}}} M_{\text{Ra}}^A(T). \quad (11.14)$$

We impose lower and upper bounds on all external and internal flow rates,

$$0 \leq Q_{\text{Ra}}, Q_{\text{De}}, Q_{\text{Ex}}, Q_{\text{Fe}}, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6 \leq Q_{\max} = 2. \quad (11.15)$$

To avoid draining inflow into outflow streams without going through a column,

$$Q_i - w_{i\text{De}} Q_{\text{De}} - w_{i\text{Fe}} Q_{\text{Fe}} >= 0 \quad (11.16)$$

has to hold for all $i \in I$. The objective is to maximize the feed throughput $M_{\text{Fe}}(T)/T$. Summarizing, we obtain the following MIOCP

$$\begin{aligned}
& \max_{x(\cdot), u(\cdot), w(\cdot), p, T} M_{\text{Fe}}(T)/T \\
\text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t), w(t), p), \\
& x(0) = Px(T), \\
& \sum_{i \in I} w_{i\alpha}(t) = 1, \quad \forall t \in [0, T], \\
& w(t) \in \{0, 1\}^{4N_{\text{col}}}, \quad \forall t \in [0, T].
\end{aligned} \tag{11.13 – 11.16}$$

with $\alpha \in \{\text{Ra, De, Ex, Fe}\}$.

11.2. Results. We optimized different operation schemes that fit into the general problem formulation (11.17): **SMB fix.** The $w_{i\alpha}$ are fixed as shown in [Table 7](#). The flow rates Q_i are constant in time, i.e., they enter as optimization parameters p into (11.17). Optimal solution $\Phi = 0.7345$.

SMB relaxed. As above. But the $w_{i\alpha}$ are free for optimization and relaxed to $w_{i\alpha} \in [0, 1]$, allowing for a "splitting" of the ports. $\Phi = 0.8747$. In **PowerFeed** the flow rates are modulated during one period, i.e., the Q_i enter as control functions $u(\cdot)$ into (11.17). $\Phi = 0.8452$. **VARICOL.** The ports switch asynchronously, but in a given order. The switching times are subject to optimization. $\Phi = 0.9308$. **Superstruct.** This scheme is the most general and allows for arbitrary switching of the ports. The flow rates enter as continuous control functions, but are found to be bang–bang by the optimizer (i.e., whenever the port is given in [Table 7](#), the respective flow rate is at its upper bound). $\Phi = 1.0154$.

TABLE 7

Fixed or optimized port assignment $w_{i\alpha}$ and switching times of the process strategies.

Process	Time	1	2	3	4	5	6
SMB fix	0.00 – 0.63	De	Ex	Fe	Ra		
SMB relaxed	0.00 – 0.50	De,Ex	Ex	Fe	Ra		
PowerFeed	0.00 – 0.56	De	Ex	Fe	Ra		
VARICOL	0.00 – 0.18	De	Ex	Fe	Ra		
	0.18 – 0.36	De		Ex	Fe	Ra	
	0.36 – 0.46	De,Ra		Ex	Fe		
	0.46 – 0.53	De,Ra		Ex		Fe	
Superstruct	0.00 – 0.10	Ex				De	
	0.10 – 0.18		De,Ex				
	0.18 – 0.24	De				Ra	
	0.24 – 0.49	De		Ex	Fe		Ra
	0.49 – 0.49		De,Ex				

12. Discretizations to MINLPs. In this section we provide AMPL code for two discretized variants of the control problems from Sections 3

and 4 as an illustration of the discretization of MIOCPs to MINLPs. More examples will be collected in the future on <http://mintoc.de>.

12.1. General AMPL code. In Listings 1 and 2 we provide two AMPL input files that can be included for MIOCPs with one binary control $w(t)$.

LISTING 1

Generic settings AMPL model file to be included

```
param T      > 0;      # End time
param nt     > 0;      # Number of discretization points in time
param nu     > 0;      # Number of control discretization points
param nx     > 0;      # Dimension of differential state vector
param ntpreu > 0;      # nt / nu
set I:= 0..nt;
set U:= 0..nu-1;
param uidx {I}; param fix_w; param fix_dt;

var w {U} >= 0, <= 1 binary;      # control function
var dt {U} >= 0, <= T;           # stage length vector
```

LISTING 2

Generic settings AMPL data file to be included

```
if ( fix_w > 0 ) then { for {i in U} { fix w[i]; } }
if ( fix_dt > 0 ) then { for {i in U} { fix dt[i]; } }

# Set indices of controls corresponding to time points
for {i in 0..nu-1} {
  for {j in 0..ntperu-1} { let uidx[i*ntperu+j] := i; }
}
let uidx[nt] := nu-1;
```

12.2. Lotka Volterra fishing problem. The AMPL code in Listings 3 and 4 shows a discretization of the problem(4.1) with piecewise constant controls on an equidistant grid of length T/n_u and with an implicit Euler method. Note that for other MIOCPs, especially for unstable ones as in Section 7, more advanced integration methods such as Backward Differentiation Formulae need to be applied.

LISTING 3

AMPL model file for Lotka Volterra Fishing Problem

```
var x {I, 1..nx} >= 0;
param c1 > 0; param c2 > 0; param ref1 > 0; param ref2 > 0;

minimize Deviation:
  0.5 * (dt[0]/ntperu) * ((x[0,1]-ref1)^2 + (x[0,2]-ref2)^2)
+ 0.5 * (dt[nu-1]/ntperu) * ((x[nu-1]-ref1)^2 + (x[nu-2]-ref2)^2)
+ sum {i in I diff {0,nu}} ( (dt[uidx[i]]/ntperu) *
  ( (x[i,1]-ref1)^2 + (x[i,2]-ref2)^2 ) );

subj to ODE_DISC_1 {i in I diff {0}}:
  x[i,1] = x[i-1,1] + (dt[uidx[i]]/ntperu) *
  ( x[i,1] - x[i,1]*x[i,2] - x[i,1]*c1*w[uidx[i]] );

subj to ODE_DISC_2 {i in I diff {0}}:
  x[i,2] = x[i-1,2] + (dt[uidx[i]]/ntperu) *
  ( -x[i,2] + x[i,1]*x[i,2] - x[i,2]*c2*w[uidx[i]] );

subj to overall_stage_length:
  sum {i in U} dt[i] = T;
```

LISTING 4

AMPL dat file for Lotka Volterra Fishing Problem

```

# Algorithmic parameters
param ntpetu := 100; param nu := 100; param nt := 10000;
param nx := 2;           param fix_w := 0; param fix_dt := 1;

# Problem parameters
param T := 12.0;         param c1 := 0.4;  param c2 := 0.2;
param ref1 := 1.0;        param ref2 := 1.0;

# Initial values differential states
let x[0,1] := 0.5; let x[0,2] := 0.7;
fix x[0,1]; fix x[0,2];

# Initial values control
let {i in U} w[i] := 0.0;
for {i in 0..(nu-1) / 2} { let w[i*2] := 1.0; }
let {i in U} dt[i] := T / nu;

```

Note that the constraint `overall_stage_length` is only necessary, when the value for `fix_dt` is zero, a switching time optimization.

The solution calculated by `Bonmin` (subversion revision number 1453, default settings, 3 GHz, Linux 2.6.28-13-generic, with ASL(20081205)) has an objective function value of $\Phi = 1.34434$, while the optimum of the relaxation is $\Phi = 1.3423368$. `Bonmin` needs 35301 iterations and 2741 nodes (4899.97 seconds). The intervals on the equidistant grid on which $w(t) = 1$ holds, counting from 0 to 99, are 20–32, 34, 36, 38, 40, 44, 53.

12.3. F-8 flight control. The main difficulty in calculating a time-optimal solution for the problem in Section 3 is the determination of the correct switching structure and of the switching points. If we want to formulate a MINLP, we have to slightly modify this problem. Our aim is not a minimization of the overall time, but now we want to get as close as possible to the origin $(0,0,0)$ in a prespecified time $t_f = 3.78086$ on *an equidistant time grid*. As this time grid is not a superset of the one used for the time-optimal solution in Section 3, one can not expect to reach the target state exactly. Listings 5 and 6 show the AMPL code.

LISTING 5

AMPL model file for F-8 Flight Control Problem

```

var x {I, 1..nx};
param xi > 0;

minimize Deviation: sum {i in 1..3} x[nt,i]*x[nt,i];

subj to ODE_DISC_1 {i in I diff {0}}:
x[i,1] = x[i-1,1] + (dt[uidx[i]]/ntpelu) * (
- 0.877*x[i,1] + x[i,3] - 0.088*x[i,1]*x[i,3] + 0.47*x[i,1]*x[i,1]
- 0.019*x[i,2]*x[i,2]
- x[i,1]*x[i,1]*x[i,3] + 3.846*x[i,1]*x[i,1]*x[i,1]
+ 0.215*xi - 0.28*x[i,1]*x[i,1]*xi + 0.47*x[i,1]*xi^2 - 0.63*xi^2
- 2*w[uidx[i]] * (0.215*xi - 0.28*x[i,1]*x[i,1]*xi - 0.63*xi^3));

subj to ODE_DISC_2 {i in I diff {0}}:
x[i,2] = x[i-1,2] + (dt[uidx[i]]/ntpelu) * x[i,3];

subj to ODE_DISC_3 {i in I diff {0}}:
x[i,3] = x[i-1,3] + (dt[uidx[i]]/ntpelu) * (

```

```

- 4.208*x[i,1] - 0.396*x[i,3] - 0.47*x[i,1]*x[i,1]
- 3.564*x[i,1]*x[i,1]*x[i,1]
+ 20.967*xi - 6.265*x[i,1]*x[i,1]*xi + 46*x[i,1]*xi^2 - 61.4*xi^3
- 2*w[uidx[i]]*(20.967*xi - 6.265*x[i,1]*x[i,1]*xi - 61.4*xi^3));

```

LISTING 6
AMPL dat file for F-8 Flight Control Problem

```

# Parameters
param ntpetu := 500; param nu := 60; param nt := 30000;
param nx := 3; param fix_w := 0; param fix_dt := 1;
param xi := 0.05236; param T := 8;

# Initial values differential states
let x[0,1] := 0.4655;
let x[0,2] := 0.0;
let x[0,3] := 0.0;
for {i in 1..3} { fix x[0,i]; }

# Initial values control
let {i in U} w[i] := 0.0;
for {i in 0..(nu-1) / 2} { let w[i*2] := 1.0; }
let {i in U} dt[i] := 3.78086 / nu;

```

The solution calculated by **Bonmin** has an objective function value of $\Phi = 0.023405$, while the optimum of the relaxation is $\Phi = 0.023079$. **Bonmin** needs 85702 iterations and 7031 nodes (64282 seconds). The intervals on the equidistant grid on which $w(t) = 1$ holds, counting from 0 to 59, are 0, 1, 31, 32, 42, 52, and 54. This optimal solution is shown in Figure 10.

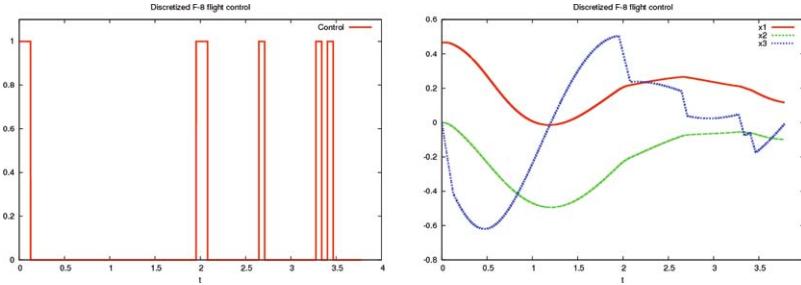


FIG. 10. *Trajectories for the discretized F-8 flight control problem. Left: optimal integer control. Right: corresponding differential states.*

13. Conclusions and outlook. We presented a collection of mixed-integer optimal control problem descriptions. These descriptions comprise details on the model and a specific instance of control objective, constraints, parameters, and initial values that yield well-posed optimization problems that allow for reproducibility and comparison of solutions. Furthermore, specific discretizations in time and space are applied with the intention to supply benchmark problems also for MINLP algorithm developers. The descriptions are complemented by references and best known solutions. All problem formulations are or will be available for download at <http://mintoc.de> in a suited format, such as **optimica** or **AMPL**.

The author hopes to achieve at least two things. First, to provide a benchmark library that will be of use for both MIOC and MINLP algorithm developers. Second, to motivate others to contribute to the extension of this library. For example, challenging and well-posed instances from water or gas networks [11, 45], traffic flow [31, 22], supply chain networks [27], submarine control [51], distributed autonomous systems [1], and chemical engineering [35, 60] would be highly interesting for the community.

Acknowledgements. Important contributions to the online resource <http://mintoc.de> by Alexander Buchner, Michael Engelhart, Christian Kirches, and Martin Schlüter are gratefully acknowledged.

REFERENCES

- [1] P. ABICHANDANI, H. BENSON, AND M. KAM, *Multi-vehicle path coordination under communication constraints*, in American Control Conference, 2008, pp. 650–656.
- [2] W. ACHTZIGER AND C. KANZOW, *Mathematical programs with vanishing constraints: optimality conditions and constraint qualifications*, Mathematical Programming Series A, **114** (2008), pp. 69–99.
- [3] E.S. AGENCY, *GTOP database: Global optimisation trajectory problems and solutions*. <http://www.esa.int/gsp/ACT/inf/op/globopt.htm>.
- [4] AT&T Bell Laboratories, University of Tennessee, and Oak Ridge National Laboratory, *Netlib linear programming library*. <http://www.netlib.org/lp/>.
- [5] B. BAUMRUCKER AND L. BIEGLER, *MPEC strategies for optimization of a class of hybrid dynamic systems*, Journal of Process Control, **19** (2009), pp. 1248 – 1256. Special Section on Hybrid Systems: Modeling, Simulation and Optimization.
- [6] B. BAUMRUCKER, J. RENFRO, AND L. BIEGLER, *MPEC problem formulations and solution strategies with chemical engineering applications*, Computers and Chemical Engineering, **32** (2008), pp. 2903–2913.
- [7] L. BIEGLER, *Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation*, Computers and Chemical Engineering, **8** (1984), pp. 243–248.
- [8] T. BINDER, L. BLANK, H. BOCK, R. BULIRSCH, W. DAHMEN, M. DIEHL, T. KRONSEDER, W. MARQUARDT, J. SCHLÖDER, AND O. STRYK, *Introduction to model based optimization of chemical processes on moving horizons*, in Online Optimization of Large Scale Systems: State of the Art, M. Grötschel, S. Krumke, and J. Rambau, eds., Springer, 2001, pp. 295–340.
- [9] H. BOCK AND R. LONGMAN, *Computation of optimal controls on disjoint control sets for minimum energy subway operation*, in Proceedings of the American Astronomical Society. Symposium on Engineering Science and Mechanics, Taiwan, 1982.
- [10] H. BOCK AND K. PLITT, *A Multiple Shooting algorithm for direct solution of optimal control problems*, in Proceedings of the 9th IFAC World Congress, Budapest, 1984, Pergamon Press, pp. 243–247. Available at <http://www.iwr.uni-heidelberg.de/groups/agbock/FILES/Bock1984.pdf>.
- [11] J. BURGSCHWEIGER, B. GNÄDIG, AND M. STEINBACH, *Nonlinear programming techniques for operative planning in large drinking water networks*, The Open Applied Mathematics Journal, **3** (2009), pp. 1–16.
- [12] M. BUSSIECK, *Gams performance world*. <http://www.gamsworld.org/performance>.

- [13] M.R. BUSSIECK, A.S. DRUD, AND A. MEERAUS, *Minlplib—a collection of test models for mixed-integer nonlinear programming*, INFORMS J. on Computing, **15** (2003), pp. 114–119.
- [14] B. CHACHUAT, A. SINGER, AND P. BARTON, *Global methods for dynamic optimization and mixed-integer dynamic optimization*, Industrial and Engineering Chemistry Research, **45** (2006), pp. 8573–8392.
- [15] CMU-IBM, *Cyber-infrastructure for MINLP collaborative site*. <http://minlp.org>.
- [16] M. DIEHL AND A. WALTHER, *A test problem for periodic optimal control algorithms*, tech. rep., ESAT/SISTA, K.U. Leuven, 2006.
- [17] S. ENGELL AND A. TOUMI, *Optimisation and control of chromatography*, Computers and Chemical Engineering, **29** (2005), pp. 1243–1252.
- [18] W. ESPOSITO AND C. FLOUDAS, *Deterministic global optimization in optimal control problems*, Journal of Global Optimization, **17** (2000), pp. 97–126.
- [19] EUROPEAN NETWORK OF EXCELLENCE HYBRID CONTROL, *Website*. <http://www.ist-hycon.org/>.
- [20] B.C. FABIEN, *dsoa: Dynamic system optimization*. <http://abs-5.me.washington.edu/noc/dsoa.html>.
- [21] A. FILIPPOV, *Differential equations with discontinuous right hand side*, AMS Transl., **42** (1964), pp. 199–231.
- [22] A. FÜGENSCHUH, M. HERTY, A. KLAR, AND A. MARTIN, *Combinatorial and continuous models for the optimization of traffic flows on networks*, SIAM Journal on Optimization, **16** (2006), pp. 1155–1176.
- [23] A. FULLER, *Study of an optimum nonlinear control system*, Journal of Electronics and Control, **15** (1963), pp. 63–71.
- [24] W. GARRARD AND J. JORDAN, *Design of nonlinear automatic control systems*, Automatica, **13** (1977), pp. 497–505.
- [25] M. GERDTS, *Solving mixed-integer optimal control problems by Branch&Bound: A case study from automobile test-driving with gear shift*, Optimal Control Applications and Methods, **26** (2005), pp. 1–18.
- [26] ———, *A variable time transformation method for mixed-integer optimal control problems*, Optimal Control Applications and Methods, **27** (2006), pp. 169–182.
- [27] S. GÖTTLICH, M. HERTY, C. KIRCHNER, AND A. KLAR, *Optimal control for continuous supply network models*, Networks and Heterogenous Media, **1** (2007), pp. 675–688.
- [28] N. GOULD, D. ORBAN, AND P. TOINT, *CUTER testing environment for optimization and linear algebra solvers*. <http://cuter.rl.ac.uk/cuter-www/>.
- [29] I. GROSSMANN, *Review of nonlinear mixed-integer and disjunctive programming techniques*, Optimization and Engineering, **3** (2002), pp. 227–252.
- [30] I. GROSSMANN, P. AGUIRRE, AND M. BARTTFELD, *Optimal synthesis of complex distillation columns using rigorous models*, Computers and Chemical Engineering, **29** (2005), pp. 1203–1215.
- [31] M. GUGAT, M. HERTY, A. KLAR, AND G. LEUGERING, *Optimal control for traffic flow networks*, Journal of Optimization Theory and Applications, **126** (2005), pp. 589–616.
- [32] T.O. INC., *Propet - matlab optimal control software (dae, ode)*. <http://tomdyn.com/>.
- [33] A. IZMAILOV AND M. SOLODOV, *Mathematical programs with vanishing constraints: Optimality conditions, sensitivity, and a relaxation method*, Journal of Optimization Theory and Applications, **142** (2009), pp. 501–532.
- [34] Y. KAWAJIRI AND L. BIEGLER, *A nonlinear programming superstructure for optimal dynamic operations of simulated moving bed processes*, I&EC Research, **45** (2006), pp. 8503–8513.
- [35] ———, *Optimization strategies for Simulated Moving Bed and PowerFeed processes*, AIChE Journal, **52** (2006), pp. 1343–1350.
- [36] C. KAYA AND J. NOAKES, *A computational method for time-optimal control*, Journal of Optimization Theory and Applications, **117** (2003), pp. 69–92.

- [37] C. KIRCHES, S. SAGER, H. BOCK, AND J. SCHLÖDER, *Time-optimal control of automobile test drives with gear shifts*, Optimal Control Applications and Methods, **31** (2010), pp. 137–153.
- [38] P. KRÄMER-EIS, *Ein Mehrzielverfahren zur numerischen Berechnung optimaler Feedback-Steuerungen bei beschränkten nichtlinearen Steuerungsproblemen*, Vol. **166** of Bonner Mathematische Schriften, Universität Bonn, Bonn, 1985.
- [39] U. KUMMER, L. OLSEN, C. DIXON, A. GREEN, E. BORNBERG-BAUER, AND G. BAIER, *Switching from simple to complex oscillations in calcium signalling*, Biophysical Journal, **79** (2000), pp. 1188–1195.
- [40] L. LARSEN, R. IZADI-ZAMANABADI, R. WISNIEWSKI, AND C. SONNTAG, *Supermarket refrigeration systems – a benchmark for the optimal control of hybrid systems*, tech. rep., Technical report for the HYCON NoE., 2007. <http://www.bci.tu-dortmund.de/ast/hycon4b/index.php>.
- [41] D. LEBIEDZ, S. SAGER, H. BOCK, AND P. LEBIEDZ, *Annihilation of limit cycle oscillations by identification of critical phase resetting stimuli via mixed-integer optimal control methods*, Physical Review Letters, **95** (2005), p. 108303.
- [42] H. LEE, K. TEO, V. REHBOCK, AND L. JENNINGS, *Control parametrization enhancing technique for time-optimal control problems*, Dynamic Systems and Applications, **6** (1997), pp. 243–262.
- [43] D. LEINEWEBER, *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, Vol. **613** of Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik, VDI Verlag, Düsseldorf, 1999.
- [44] A. MARTIN, T. ACHTERBERG, T. KOCH, AND G. GAMRATH, *Miplib - mixed integer problem library*. <http://miplib.zib.de/>.
- [45] A. MARTIN, M. MÖLLER, AND S. MORITZ, *Mixed integer models for the stationary case of gas network optimization*, Mathematical Programming, **105** (2006), pp. 563–582.
- [46] J. OLDENBURG, *Logic-based modeling and optimization of discrete-continuous dynamic systems*, Vol. **830** of Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik, VDI Verlag, Düsseldorf, 2005.
- [47] J. OLDENBURG AND W. MARQUARDT, *Disjunctive modeling for optimal control of hybrid systems*, Computers and Chemical Engineering, **32** (2008), pp. 2346–2364.
- [48] I. PAPAMICHAIL AND C. ADJIMAN, *Global optimization of dynamic systems*, Computers and Chemical Engineering, **28** (2004), pp. 403–415.
- [49] L. PONTRYAGIN, V. BOLTYANSKI, R. GAMKRELIDZE, AND E. MISCELENKO, *The Mathematical Theory of Optimal Processes*, Wiley, Chichester, 1962.
- [50] A. PRATA, J. OLDENBURG, A. KROLL, AND W. MARQUARDT, *Integrated scheduling and dynamic optimization of grade transitions for a continuous polymerization reactor*, Computers and Chemical Engineering, **32** (2008), pp. 463–476.
- [51] V. REHBOCK AND L. CACCIETTA, *Two defence applications involving discrete valued optimal control*, ANZIAM Journal, **44** (2002), pp. E33–E54.
- [52] S. SAGER, *MIOCP benchmark site*. <http://mintoc.de>.
- [53] S. SAGER, *Numerical methods for mixed-integer optimal control problems*, Der andere Verlag, Tönning, Lübeck, Marburg, 2005. ISBN 3-89959-416-9. Available at <http://sager1.de/sebastian/downloads/Sager2005.pdf>.
- [54] S. SAGER, *Reformulations and algorithms for the optimization of switching decisions in nonlinear optimal control*, Journal of Process Control, **19** (2009), pp. 1238–1247.
- [55] S. SAGER, H. BOCK, M. DIEHL, G. REINELT, AND J. SCHLÖDER, *Numerical methods for optimal control with binary control functions applied to a Lotka-Volterra type fishing problem*, in Recent Advances in Optimization (Proceedings of the 12th French-German-Spanish Conference on Optimization), A. Seeger, ed., Vol. **563** of Lectures Notes in Economics and Mathematical Systems, Heidelberg, 2006, Springer, pp. 269–289.

- [56] S. SAGER, M. DIEHL, G. SINGH, A. KÜPPER, AND S. ENGELL, *Determining SMB superstructures by mixed-integer control*, in Proceedings OR2006, K.-H. Waldmann and U. Stocker, eds., Karlsruhe, 2007, Springer, pp. 37–44.
- [57] S. SAGER, C. KIRCHES, AND H. BOCK, *Fast solution of periodic optimal control problems in automobile test-driving with gear shifts*, in Proceedings of the 47th IEEE Conference on Decision and Control (CDC 2008), Cancun, Mexico, 2008, pp. 1563–1568. ISBN: 978-1-4244-3124-3.
- [58] S. SAGER, G. REINELT, AND H. BOCK, *Direct methods with maximal lower bound for mixed-integer optimal control problems*, Mathematical Programming, **118** (2009), pp. 109–149.
- [59] K. SCHITTKOWSKI, *Test problems for nonlinear programming - user's guide*, tech. rep., Department of Mathematics, University of Bayreuth, 2002.
- [60] C. SONNTAG, O. STURSBERG, AND S. ENGELL, *Dynamic optimization of an industrial evaporator using graph search with embedded nonlinear programming*, in Proc. 2nd IFAC Conf. on Analysis and Design of Hybrid Systems (ADHS), 2006, pp. 211–216.
- [61] B. SRINIVASAN, S. PALANKI, AND D. BONVIN, *Dynamic Optimization of Batch Processes: I. Characterization of the nominal solution*, Computers and Chemical Engineering, **27** (2003), pp. 1–26.
- [62] M. SZYMKAT AND A. KORYTOWSKI, *The method of monotone structural evolution for dynamic optimization of switched systems*, in IEEE CDC08 Proceedings, 2008.
- [63] M. ZELIKIN AND V. BORISOV, *Theory of chattering control with applications to astronautics, robotics, economics and engineering*, Birkhäuser, Basel Boston Berlin, 1994.

IMA HOT TOPICS WORKSHOP PARTICIPANTS

Mixed-Integer Nonlinear Optimization: Algorithmic Advances and Applications

- Kurt M. Anstreicher, Department of Management Sciences, University of Iowa
- Pietro Belotti, Department of Mathematical Sciences, Clemson University
- Hande Yurttan Benson, Department of Decision Sciences, Drexel University
- Dimitris Bertsimas, Sloan School of Management, Massachusetts Institute of Technology
- Lorenz T. Biegler, Chemical Engineering Department, Carnegie Mellon University
- Christian Blieck, CPLEX R&D, ILOG Corporation
- Pierre Bonami, Laboratoire d'Informatique Fondamentale de Marseille, Centre National de la Recherche Scientifique (CNRS)
- Samuel Burer, Department of Management Sciences, University of Iowa
- Alfonso Cano, University of Minnesota
- Xianjin Chen, Institute for Mathematics and its Applications, University of Minnesota
- Claudia D'Ambrosio, Dipartimento di Elettronica, Informatica e Sistemistica, Universit di Bologna
- Michel Jacques Daydé ENSEEIHT, Institut National Polytechnique de Toulouse
- Jesus Antonio De Loera, Department of Mathematics, University of California, Davis
- Sarah Drewes, Mathematics, Research Group Nonlinear Optimization, Technische Universität Darmstadt
- Ricardo Fukasawa, T.J. Watson Research Center, IBM
- Kevin Furman, Exxon Research and Engineering Company
- Weigu Gao, School of Mathematical Sciences, Fudan University
- David M. Gay, Optimization and Uncertainty Estimation, Sandia National Laboratories
- Philip E. Gill, Department of Mathematics, University of California, San Diego
- Ignacio E. Grossmann, Department of Chemical Engineering, Carnegie Mellon University

- Oktay Gunluk, Mathematical Sciences Department, IBM
- William E. Hart, Sandia National Laboratories
- David Haws, Department of Mathematics, University of California, Davis
- Christoph Helmberg, Fakultät für Mathematik, Technische Universität Chemnitz-Zwickau
- Andreas G. Karabis, Department of Research and Development, PI Medical Ltd.
- Markus Keel, Institute for Mathematics and its Applications, University of Minnesota
- Vassilio kekatos, University of Minnesota
- Mustafa Rasim Kilinc, Department of Industrial Engineering, University of Pittsburgh
- Erica Zimmer Klampfl, Ford Research Laboratory, Ford
- Matthias Koeppe, Department of Mathematics, University of California, Davis
- Jon Lee, Mathematical Sciences Department, IBM T.J. Watson Research Center
- Thomas Lehmann, Fachgruppe Informatik, Universität Bayreuth
- Sven Leyffer, Mathematics and Computer Science Division, Argonne National Laboratory
- Tong Li, Department of Mathematics, University of Iowa
- Yongfeng Li Institute for Mathematics and its Applications University of Minnesota
- Leo Liberti, LIX, École Polytechnique
- Jeff Linderoth, Department of Industrial and Systems Engineering, University of Wisconsin-Madison
- Chun Liu, Institute for Mathematics and its Applications, University of Minnesota
- Andrea Lodi, DEIS, Universitá di Bologna
- James Luedtke, Industrial and Systems Engineering Department, University of Wisconsin-Madison
- Tom Luo, Department of Electrical and Computer Engineering, University of Minnesota
- Francois Margot, Tepper School of Business, Carnegie Mellon University
- Susan Margulies, Department of Computational and Applied Mathematics, Rice University
- Andrew James Miller, Department of Mathematics, Université de Bordeaux I

- Kien Ming Ng, Department of Industrial and Systems Engineering, National University of Singapore
- John E. Mitchell, Department of Mathematical Sciences, Rensselaer Polytechnic Institute
- Hans Mittelmann, Department of Mathematics and Statistics, Arizona State University
- Todd S. Munson, Mathematics and Computer Science Division, Argonne National Laboratory
- Mahdi Namazifar, Industrial and Systems Engineering Department, University of Wisconsin-Madison
- Giacomo Nannicini, Laboratoire d'Informatique École Polytechnique
- Jorge Nocedal, Department of Electrical Engineering and Computer Science, Northwestern University
- Isamu Onishi, Department of Mathematical and Life Sciences, Hiroshima University
- Shmuel Onn, IE & M Technion-Israel Institute of Technology
- Pablo A. Parrilo, Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology
- Jaroslav Pekar, Honeywell Prague Laboratory, Honeywell
- Jiming Peng, Department of Industrial and Enterprise System Engineering, University of Illinois at Urbana-Champaign
- Cynthia A. Phillips, Discrete Mathematics and Complex Systems Department, Sandia National Laboratories
- Kashif Rashid, Schlumberger-Doll
- Franz F. Rendl, Institut für Mathematik, Universität Klagenfurt
- Kees Roos, Department of Information Systems and Algorithms, Technische Universiteit te Delft
- Sebastian Sager, Interdisciplinary Center for Scientific Computing, Ruprecht-Karls-Universität Heidelberg
- Fadil Santosa, Institute for Mathematics and its Applications, University of Minnesota
- Annick Sartenaer, Department of Mathematics, Facultés Universitaires Notre Dame de la Paix (Namur)
- Anureet Saxena, Axioma Inc.
- Uday V. Shanbhag, Industrial & Enterprise Systems Engineering Department, University of Illinois at Urbana-Champaign
- Tamás Terlaky, Department of Industrial and Systems Engineering, Lehigh University
- Jon Van Laarhoven, Department of Applied Mathematics and Computational Sciences, University of Iowa

- Stefan Vigerske, Department of Mathematics, Humboldt-Universität
- Andreas Wächter, Mathematical Sciences Department, IBM
- Richard A. Waltz, Department of Industrial and Systems Engineering, University of Southern California
- Robert Weismantel, Department of Mathematical Optimization, Otto-von-Guericke-Universität Magdeburg
- Tapio Westerlund, Department of Chemical Engineering, Åbo Akademi (Finland-Swedish University of Åbo)
- Angelika Wiegele, Department of Mathematics, Universität Klagenfurt
- Fei Yang, Department of Biomedical Engineering, University of Minnesota
- Hongchao Zhang, Department of Mathematics, Louisiana State University

- 1992–1993 Control Theory and its Applications
1993–1994 Emerging Applications of Probability
1994–1995 Waves and Scattering
1995–1996 Mathematical Methods in Material Science
1996–1997 Mathematics of High Performance Computing
1997–1998 Emerging Applications of Dynamical Systems
1998–1999 Mathematics in Biology
1999–2000 Reactive Flows and Transport Phenomena
2000–2001 Mathematics in Multimedia
2001–2002 Mathematics in the Geosciences
2002–2003 Optimization
2003–2004 Probability and Statistics in Complex Systems: Genomics, Networks, and Financial Engineering
2004–2005 Mathematics of Materials and Macromolecules: Multiple Scales, Disorder, and Singularities
2005–2006 Imaging
2006–2007 Applications of Algebraic Geometry
2007–2008 Mathematics of Molecular and Cellular Biology
2008–2009 Mathematics and Chemistry
2009–2010 Complex Fluids and Complex Flows
2010–2011 Simulating Our Complex World: Modeling, Computation and Analysis
2011–2012 Mathematics of Information
2012–2013 Infinite Dimensional and Stochastic Dynamical Systems and their Applications
2013–2014 Scientific and Engineering Applications of Algebraic Topology

IMA SUMMER PROGRAMS

- 1987 Robotics
1988 Signal Processing
1989 Robust Statistics and Diagnostics
1990 Radar and Sonar (June 18–29)
New Directions in Time Series Analysis (July 2–27)
1991 Semiconductors
1992 Environmental Studies: Mathematical, Computational, and Statistical Analysis
1993 Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations
1994 Molecular Biology

- 1995 Large Scale Optimizations with Applications to Inverse Problems, Optimal Control and Design, and Molecular and Structural Optimization
- 1996 Emerging Applications of Number Theory (July 15–26)
Theory of Random Sets (August 22–24)
- 1997 Statistics in the Health Sciences
- 1998 Coding and Cryptography (July 6–18)
Mathematical Modeling in Industry (July 22–31)
- 1999 Codes, Systems, and Graphical Models (August 2–13, 1999)
- 2000 Mathematical Modeling in Industry: A Workshop for Graduate Students (July 19–28)
- 2001 Geometric Methods in Inverse Problems and PDE Control (July 16–27)
- 2002 Special Functions in the Digital Age (July 22–August 2)
- 2003 Probability and Partial Differential Equations in Modern Applied Mathematics (July 21–August 1)
- 2004 n-Categories: Foundations and Applications (June 7–18)
- 2005 Wireless Communications (June 22–July 1)
- 2006 Symmetries and Overdetermined Systems of Partial Differential Equations (July 17–August 4)
- 2007 Classical and Quantum Approaches in Molecular Modeling (July 23–August 3)
- 2008 Geometrical Singularities and Singular Geometries (July 14–25)
- 2009 Nonlinear Conservation Laws and Applications (July 13–31)

IMA "HOT TOPICS/SPECIAL" WORKSHOPS

- Challenges and Opportunities in Genomics: Production, Storage, Mining and Use, April 24–27, 1999
- Decision Making Under Uncertainty: Energy and Environmental Models, July 20–24, 1999
- Analysis and Modeling of Optical Devices, September 9–10, 1999
- Decision Making under Uncertainty: Assessment of the Reliability of Mathematical Models, September 16–17, 1999
- Scaling Phenomena in Communication Networks, October 22–24, 1999
- Text Mining, April 17–18, 2000
- Mathematical Challenges in Global Positioning Systems (GPS), August 16–18, 2000
- Modeling and Analysis of Noise in Integrated Circuits and Systems, August 29–30, 2000
- Mathematics of the Internet: E-Auction and Markets, December 3–5, 2000
- Analysis and Modeling of Industrial Jetting Processes, January 10–13, 2001

- Special Workshop: Mathematical Opportunities in Large-Scale Network Dynamics, August 6–7, 2001
- Wireless Networks, August 8–10 2001
- Numerical Relativity, June 24–29, 2002
- Operational Modeling and Biodefense: Problems, Techniques, and Opportunities, September 28, 2002
- Data-driven Control and Optimization, December 4–6, 2002
- Agent Based Modeling and Simulation, November 3–6, 2003
- Enhancing the Search of Mathematics, April 26–27, 2004
- Compatible Spatial Discretizations for Partial Differential Equations, May 11–15, 2004
- Adaptive Sensing and Multimode Data Inversion, June 27–30, 2004
- Mixed Integer Programming, July 25–29, 2005
- New Directions in Probability Theory, August 5–6, 2005
- Negative Index Materials, October 2–4, 2006
- The Evolution of Mathematical Communication in the Age of Digital Libraries, December 8–9, 2006
- Math is Cool! and Who Wants to Be a Mathematician?, November 3, 2006
- Special Workshop: Blackwell-Tapia Conference, November 3–4, 2006
- Stochastic Models for Intracellular Reaction Networks, May 11–13, 2008
- Multi-Manifold Data Modeling and Applications, October 27–30, 2008
- Mixed-Integer Nonlinear Optimization: Algorithmic Advances and Applications, November 17–21, 2008
- Higher Order Geometric Evolution Equations: Theory and Applications from Microfluidics to Image Understanding, March 23–26, 2009
- Career Options for Women in Mathematical Sciences, April 2–4, 2009
- MOLCAS, May 4–8, 2009
- IMA Interdisciplinary Research Experience for Undergraduates, June 29–July 31, 2009
- Research in Imaging Sciences, October 5–7, 2009
- Career Options for Underrepresented Groups in Mathematical Sciences, March 25–27, 2010
- Physical Knotting and Linking and its Applications, April 9, 2010
- IMA Interdisciplinary Research Experience for Undergraduates, June 14–July 16, 2010
- Kickoff Workshop for Project MOSAIC, June 30–July 2, 2010
- Finite Element Circus Featuring a Scientific Celebration of Falk, Pasciak, and Wahlbin, November 5–6, 2010

- Integral Equation Methods, Fast Algorithms and Applications, August 2–5, 2010
- Medical Device-Biological Interactions at the Material-Tissue Interface, September 13–15, 2010
- First Abel Conference A Mathematical Celebration of John Tate, January 3–5, 2011
- Strain Induced Shape Formation: Analysis, Geometry and Materials Science, May 16–20, 2011
- Uncertainty Quantification in Industrial and Energy Applications: Experiences and Challenges, June 2–4, 2011
- Girls and Mathematics Summer Day Program, June 20–24, 2011
- Special Workshop: Wavelets and Applications: A Multi-Disciplinary Undergraduate Course with an Emphasis on Scientific Computing, July 13–16, 2011
- Special Workshop: Wavelets and Applications: Project Building Workshop, July 13–16, 2011
- Macaulay2, July 25–29, 2011
- Instantaneous Frequencies and Trends for Nonstationary Nonlinear Data, September 7–9, 2011

SPRINGER LECTURE NOTES FROM THE IMA

The Mathematics and Physics of Disordered Media

Editors: Barry Hughes and Barry Ninham
(Lecture Notes in Math., Volume 1035, 1983)

Orienting Polymers

Editor: J.L. Ericksen
(Lecture Notes in Math., Volume 1063, 1984)

New Perspectives in Thermodynamics

Editor: James Serrin
(Springer-Verlag, 1986)

Models of Economic Dynamics

Editor: Hugo Sonnenschein
(Lecture Notes in Econ., Volume 264, 1986)

**THE IMA VOLUMES
IN MATHEMATICS AND ITS APPLICATIONS**

Volume 1: Homogenization and Effective Moduli of Materials and Media

Editors: Jerry Ericksen, David Kinderlehrer, Robert Kohn, and J.-L. Lions

Volume 2: Oscillation Theory, Computation, and Methods of Compensated Compactness

Editors: Constantine Dafermos, Jerry Ericksen, David Kinderlehrer, and Marshall Slemrod

Volume 3: Metastability and Incompletely Posed Problems

Editors: Stuart Antman, Jerry Ericksen, David Kinderlehrer, and Ingo Muller

Volume 4: Dynamical Problems in Continuum Physics

Editors: Jerry Bona, Constantine Dafermos, Jerry Ericksen, and David Kinderlehrer

Volume 5: Theory and Applications of Liquid Crystals

Editors: Jerry Ericksen and David Kinderlehrer

Volume 6: Amorphous Polymers and Non-Newtonian Fluids

Editors: Constantine Dafermos, Jerry Ericksen, and David Kinderlehrer

Volume 7: Random Media

Editor: George Papanicolaou

Volume 8: Percolation Theory and Ergodic Theory of Infinite Particle Systems

Editor: Harry Kesten

Volume 9: Hydrodynamic Behavior and Interacting Particle Systems

Editor: George Papanicolaou

Volume 10: Stochastic Differential Systems, Stochastic Control Theory, and Applications

Editors: Wendell Fleming and Pierre-Louis Lions

Volume 11: Numerical Simulation in Oil Recovery

Editor: Mary Fanett Wheeler

Volume 12: Computational Fluid Dynamics and Reacting Gas Flows
Editors: Bjorn Engquist, M. Luskin, and Andrew Majda

Volume 13: Numerical Algorithms for Parallel Computer Architectures
Editor: Martin H. Schultz

Volume 14: Mathematical Aspects of Scientific Software
Editor: J.R. Rice

Volume 15: Mathematical Frontiers in Computational Chemical Physics
Editor: D. Truhlar

Volume 16: Mathematics in Industrial Problems
by Avner Friedman

Volume 17: Applications of Combinatorics and Graph Theory to the
Biological and Social Sciences
Editor: Fred Roberts

Volume 18: *q*-Series and Partitions
Editor: Dennis Stanton

Volume 19: Invariant Theory and Tableaux
Editor: Dennis Stanton

Volume 20: Coding Theory and Design Theory Part I: Coding Theory
Editor: Dijen Ray-Chaudhuri

Volume 21: Coding Theory and Design Theory Part II: Design Theory
Editor: Dijen Ray-Chaudhuri

Volume 22: Signal Processing: Part I - Signal Processing Theory
Editors: L. Auslander, F.A. Grünbaum, J.W. Helton, T. Kailath,
P. Khargonekar, and S. Mitter

Volume 23: Signal Processing: Part II - Control Theory and Applications
of Signal Processing
Editors: L. Auslander, F.A. Grünbaum, J.W. Helton, T. Kailath,
P. Khargonekar, and S. Mitter

Volume 24: Mathematics in Industrial Problems, Part II
by Avner Friedman

Volume 25: Solitons in Physics, Mathematics, and Nonlinear Optics
Editors: Peter J. Olver and David H. Sattinger

Volume 26: Two Phase Flows and Waves

Editors: Daniel D. Joseph and David G. Schaeffer

Volume 27: Nonlinear Evolution Equations that Change Type

Editors: Barbara Lee Keyfitz and Michael Shearer

Volume 28: Computer Aided Proofs in Analysis

Editors: Kenneth Meyer and Dieter Schmidt

Volume 29: Multidimensional Hyperbolic Problems and Computations

Editors: Andrew Majda and Jim Glimm

Volume 30: Microlocal Analysis and Nonlinear Waves

Editors: Michael Beals, R. Melrose, and J. Rauch

Volume 31: Mathematics in Industrial Problems, Part III

by Avner Friedman

Volume 32: Radar and Sonar, Part I

by Richard Blahut, Willard Miller, Jr., and Calvin Wilcox

Volume 33: Directions in Robust Statistics and Diagnostics: Part I

Editors: Werner A. Stahel and Sanford Weisberg

Volume 34: Directions in Robust Statistics and Diagnostics: Part II

Editors: Werner A. Stahel and Sanford Weisberg

Volume 35: Dynamical Issues in Combustion Theory

Editors: P. Fife, A. Liñán, and F.A. Williams

Volume 36: Computing and Graphics in Statistics

Editors: Andreas Buja and Paul Tukey

Volume 37: Patterns and Dynamics in Reactive Media

Editors: Harry Swinney, Gus Aris, and Don Aronson

Volume 38: Mathematics in Industrial Problems, Part IV

by Avner Friedman

Volume 39: Radar and Sonar, Part II

Editors: F. Alberto Grünbaum, Marvin Bernfeld, and
Richard E. Blahut

Volume 40: Nonlinear Phenomena in Atmospheric and Oceanic Sciences

Editors: George F. Carnevale and Raymond T. Pierrehumbert

Volume 41: Chaotic Processes in the Geological Sciences
Editor: David A. Yuen

Volume 42: Partial Differential Equations with Minimal Smoothness and Applications
Editors: B. Dahlberg, E. Fabes, R. Fefferman, D. Jerison, C. Kenig, and J. Pipher

Volume 43: On the Evolution of Phase Boundaries
Editors: Morton E. Gurtin and Geoffrey B. McFadden

Volume 44: Twist Mappings and Their Applications
Editors: Richard McGehee and Kenneth R. Meyer

Volume 45: New Directions in Time Series Analysis, Part I
Editors: David Brillinger, Peter Caines, John Geweke, Emanuel Parzen, Murray Rosenblatt, and Murad S. Taqqu

Volume 46: New Directions in Time Series Analysis, Part II
Editors: David Brillinger, Peter Caines, John Geweke, Emanuel Parzen, Murray Rosenblatt, and Murad S. Taqqu

Volume 47: Degenerate Diffusions
Editors: Wei-Ming Ni, L.A. Peletier, and J.-L. Vazquez

Volume 48: Linear Algebra, Markov Chains, and Queueing Models
Editors: Carl D. Meyer and Robert J. Plemmons

Volume 49: Mathematics in Industrial Problems, Part V
by Avner Friedman

Volume 50: Combinatorial and Graph-Theoretic Problems in Linear Algebra
Editors: Richard A. Brualdi, Shmuel Friedland, and Victor Klee

Volume 51: Statistical Thermodynamics and Differential Geometry of Microstructured Materials
Editors: H. Ted Davis and Johannes C.C. Nitsche

Volume 52: Shock Induced Transitions and Phase Structures in General Media
Editors: J.E. Dunn, Roger Fosdick, and Marshall Slemrod

Volume 53: Variational and Free Boundary Problems
Editors: Avner Friedman and Joel Spruck

Volume 54: Microstructure and Phase Transitions

Editors: David Kinderlehrer, Richard James, Mitchell Luskin, and Jerry L. Ericksen

Volume 55: Turbulence in Fluid Flows: A Dynamical Systems Approach

Editors: George R. Sell, Ciprian Foias, and Roger Temam

Volume 56: Graph Theory and Sparse Matrix Computation

Editors: Alan George, John R. Gilbert, and Joseph W.H. Liu

Volume 57: Mathematics in Industrial Problems, Part VI

by Avner Friedman

Volume 58: Semiconductors, Part I

Editors: W.M. Coughran, Jr., Julian Cole, Peter Lloyd, and Jacob White

Volume 59: Semiconductors, Part II

Editors: W.M. Coughran, Jr., Julian Cole, Peter Lloyd, and Jacob White

Volume 60: Recent Advances in Iterative Methods

Editors: Gene Golub, Anne Greenbaum, and Mitchell Luskin

Volume 61: Free Boundaries in Viscous Flows

Editors: Robert A. Brown and Stephen H. Davis

Volume 62: Linear Algebra for Control Theory

Editors: Paul Van Dooren and Bostwick Wyman

Volume 63: Hamiltonian Dynamical Systems: History, Theory, and

Applications

Editors: H.S. Dumas, K.R. Meyer, and D.S. Schmidt

Volume 64: Systems and Control Theory for Power Systems

Editors: Joe H. Chow, Petar V. Kokotovic, and Robert J. Thomas

Volume 65: Mathematical Finance

Editors: Mark H.A. Davis, Darrell Duffie, Wendell H. Fleming, and Steven E. Shreve

Volume 66: Robust Control Theory

Editors: Bruce A. Francis and Pramod P. Khargonekar

Volume 67: Mathematics in Industrial Problems, Part VII
by Avner Friedman

Volume 68: Flow Control
Editor: Max D. Gunzburger

Volume 69: Linear Algebra for Signal Processing
Editors: Adam Bojanczyk and George Cybenko

Volume 70: Control and Optimal Design of Distributed Parameter Systems
Editors: John E. Lagnese, David L. Russell, and Luther W. White

Volume 71: Stochastic Networks
Editors: Frank P. Kelly and Ruth J. Williams

Volume 72: Discrete Probability and Algorithms
Editors: David Aldous, Persi Diaconis, Joel Spencer, and J. Michael Steele

Volume 73: Discrete Event Systems, Manufacturing Systems,
and Communication Networks
Editors: P.R. Kumar and P.P. Varaiya

Volume 74: Adaptive Control, Filtering, and Signal Processing
Editors: K.J. Åström, G.C. Goodwin, and P.R. Kumar

Volume 75: Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations
Editors: Ivo Babuska, Joseph E. Flaherty, William D. Henshaw, John E. Hopcroft, Joseph E. Olliger, and Tayfun Tezduyar

Volume 76: Random Discrete Structures
Editors: David Aldous and Robin Pemantle

Volume 77: Nonlinear Stochastic PDE's: Hydrodynamic Limit and Burgers' Turbulence
Editors: Tadahisa Funaki and Wojbor A. Woyczyński

Volume 78: Nonsmooth Analysis and Geometric Methods in Deterministic Optimal Control
Editors: Boris S. Mordukhovich and Hector J. Sussmann

Volume 79: Environmental Studies: Mathematical, Computational, and Statistical Analysis
Editor: Mary Fanett Wheeler

Volume 80: Image Models (and their Speech Model Cousins)
Editors: Stephen E. Levinson and Larry Shepp

Volume 81: Genetic Mapping and DNA Sequencing
Editors: Terry Speed and Michael S. Waterman

Volume 82: Mathematical Approaches to Biomolecular Structure and Dynamics
Editors: Jill P. Mesirov, Klaus Schulten, and De Witt Sumners

Volume 83: Mathematics in Industrial Problems, Part VIII
by Avner Friedman

Volume 84: Classical and Modern Branching Processes
Editors: Krishna B. Athreya and Peter Jagers

Volume 85: Stochastics Models in Geosystems
Editors: Stanislav A. Molchanov and Wojbor A. Woyczyński

Volume 86: Computational Wave Propagation
Editors: Bjorn Engquist and Gregory A. Kriegsmann

Volume 87: Progress in Population Genetics and Human Evolution
Editors: Peter Donnelly and Simon Tavaré

Volume 88: Mathematics in Industrial Problems, Part 9
by Avner Friedman

Volume 89: Multiparticle Quantum Scattering with Applications to Nuclear, Atomic and Molecular Physics
Editors: Donald G. Truhlar and Barry Simon

Volume 90: Inverse Problems in Wave Propagation
Editors: Guy Chavent, George Papanicolaou, Paul Sacks, and William Symes

Volume 91: Singularities and Oscillations
Editors: Jeffrey Rauch and Michael Taylor

Volume 92: Large-Scale Optimization with Applications, Part I: Optimization in Inverse Problems and Design
Editors: Lorenz T. Biegler, Thomas F. Coleman, Andrew R. Conn, and Fadil Santosa

Volume 93: Large-Scale Optimization with Applications, Part II: Optimal Design and Control

Editors: Lorenz T. Biegler, Thomas F. Coleman, Andrew R. Conn, and Fadil Santosa

Volume 94: Large-Scale Optimization with Applications, Part III: Molecular Structure and Optimization

Editors: Lorenz T. Biegler, Thomas F. Coleman, Andrew R. Conn, and Fadil Santosa

Volume 95: Quasiclassical Methods

Editors: Jeffrey Rauch and Barry Simon

Volume 96: Wave Propagation in Complex Media

Editor: George Papanicolaou

Volume 97: Random Sets: Theory and Applications

Editors: John Goutsias, Ronald P.S. Mahler, and Hung T. Nguyen

Volume 98: Particulate Flows: Processing and Rheology

Editors: Donald A. Drew, Daniel D. Joseph, and Stephen L. Passman

Volume 99: Mathematics of Multiscale Materials

Editors: Kenneth M. Golden, Geoffrey R. Grimmett, Richard D. James, Graeme W. Milton, and Pabitra N. Sen

Volume 100: Mathematics in Industrial Problems, Part 10

by Avner Friedman

Volume 101: Nonlinear Optical Materials

Editor: Jerome V. Moloney

Volume 102: Numerical Methods for Polymeric Systems

Editor: Stuart G. Whittington

Volume 103: Topology and Geometry in Polymer Science

Editors: Stuart G. Whittington, De Witt Sumners, and Timothy Lodge

Volume 104: Essays on Mathematical Robotics

Editors: John Baillieul, Shankar S. Sastry, and Hector J. Sussmann

Volume 105: Algorithms for Parallel Processing

Editors: Robert S. Schreiber, Michael T. Heath, and Abhiram Ranade

Volume 106: Parallel Processing of Discrete Problems

Editor: Panos Pardalos

Volume 107: The Mathematics of Information Coding, Extraction, and Distribution

Editors: George Cybenko, Dianne O'Leary, and Jorma Rissanen

Volume 108: Rational Drug Design

Editors: Donald G. Truhlar, W. Jeffrey Howe, Anthony J. Hopfinger, Jeff Blaney, and Richard A. Dammkoehler

Volume 109: Emerging Applications of Number Theory

Editors: Dennis A. Hejhal, Joel Friedman, Martin C. Gutzwiller, and Andrew M. Odlyzko

Volume 110: Computational Radiology and Imaging: Therapy and Diagnostics

Editors: Christoph Börgers and Frank Natterer

Volume 111: Evolutionary Algorithms

Editors: Lawrence David Davis, Kenneth De Jong, Michael D. Vose and L. Darrell Whitley

Volume 112: Statistics in Genetics

Editors: M. Elizabeth Halloran and Seymour Geisser

Volume 113: Grid Generation and Adaptive Algorithms

Editors: Marshall Bern, Joseph E. Flaherty, and Mitchell Luskin

Volume 114: Diagnosis and Prediction

Editor: Seymour Geisser

Volume 115: Pattern Formation in Continuous and Coupled Systems: A Survey Volume

Editors: Martin Golubitsky, Dan Luss, and Steven H. Strogatz

Volume 116: Statistical Models in Epidemiology, the Environment and Clinical Trials

Editors: M. Elizabeth Halloran and Donald Berry

Volume 117: Structured Adaptive Mesh Refinement (SAMR) Grid Methods

Editors: Scott B. Baden, Nikos P. Chrisochoides, Dennis B. Gannon, and Michael L. Norman

Volume 118: Dynamics of Algorithms

Editors: Rafael de la Llave, Linda R. Petzold, and Jens Lorenz

Volume 119: Numerical Methods for Bifurcation Problems and Large-Scale Dynamical Systems

Editors: Eusebius Doedel and Laurette S. Tuckerman

Volume 120: Parallel Solution of Partial Differential Equations

Editors: Petter Bjørstad and Mitchell Luskin

Volume 121: Mathematical Models for Biological Pattern Formation

Editors: Philip K. Maini and Hans G. Othmer

Volume 122: Multiple-Time-Scale Dynamical Systems

Editors: Christopher K.R.T. Jones and Alexander Khibnik

Volume 123: Codes, Systems, and Graphical Models

Editors: Brian Marcus and Joachim Rosenthal

Volume 124: Computational Modeling in Biological Fluid Dynamics

Editors: Lisa J. Fauci and Shay Gueron

Volume 125: Mathematical Approaches for Emerging and Reemerging Infectious Diseases: An Introduction

Editors: Carlos Castillo-Chavez with Sally Blower, Pauline van den Driessche, Denise Kirschner, and Abdul-Aziz Yakubu

Volume 126: Mathematical Approaches for Emerging and Reemerging Infectious Diseases: Models, Methods and Theory

Editors: Carlos Castillo-Chavez with Sally Blower, Pauline van den Driessche, Denise Kirschner, and Abdul-Aziz Yakubu

Volume 127: Mathematics of the Internet: E-Auction and Markets

Editors: Brenda Dietrich and Rakesh V. Vohra

Volume 128: Decision Making Under Uncertainty: Energy and Power

Editors: Claude Greengard and Andrzej Ruszcynski

Volume 129: Membrane Transport and Renal Physiology

Editors: Harold E. Layton and Alan M. Weinstein

Volume 130: Atmospheric Modeling

Editors: David P. Chock and Gregory R. Carmichael

- Volume 131: Resource Recovery, Confinement, and Remediation of Environmental Hazards
Editors: John Chadam, Al Cunningham, Richard E. Ewing, Peter Ortoleva, and Mary Fanett Wheeler
- Volume 132: Fractals in Multimedia
Editors: Michael F. Barnsley, Dietmar Saupe, and Edward Vrscay
- Volume 133: Mathematical Methods in Computer Vision
Editors: Peter J. Olver and Allen Tannenbaum
- Volume 134: Mathematical Systems Theory in Biology, Communications, Computation, and Finance
Editors: Joachim Rosenthal and David S. Gilliam
- Volume 135: Transport in Transition Regimes
Editors: Naoufel Ben Abdallah, Anton Arnold, Pierre Degond, Irene Gamba, Robert Glassey, C. David Levermore, and Christian Ringhofer
- Volume 136: Dispersive Transport Equations and Multiscale Models
Editors: Naoufel Ben Abdallah, Anton Arnold, Pierre Degond, Irene Gamba, Robert Glassey, C. David Levermore, and Christian Ringhofer
- Volume 137: Geometric Methods in Inverse Problems and PDE Control
Editors: Christopher B. Croke, Irena Lasiecka, Gunther Uhlmann, and Michael S. Vogelius
- Volume 138: Mathematical Foundations of Speech and Language Processing
Editors: Mark Johnson, Sanjeev Khudanpur, Mari Ostendorf, and Roni Rosenfeld
- Volume 139: Time Series Analysis and Applications to Geophysical Systems
Editors: David R. Brillinger, Enders Anthony Robinson, and Frederick Paik Schoenberg
- Volume 140: Probability and Partial Differential Equations in Modern Applied Mathematics
Editors: Jinqiao Duan and Edward C. Waymire
- Volume 141: Modeling of Soft Matter
Editors: Maria-Carme T. Calderer and Eugene M. Terentjev

Volume 142: Compatible Spatial Discretizations

Editors: Douglas N. Arnold, Pavel B. Bochev, Richard B. Lehoucq,
Roy A. Nicolaides, and Mikhail Shashkov

Volume 143: Wireless Communications

Editors: Prathima Agrawal, Daniel Matthew Andrews, Philip J.
Fleming, George Yin, and Lisa Zhang

Volume 144: Symmetries and Overdetermined Systems of Partial Differential Equations

Editors: Michael Eastwood and Willard Miller, Jr.

Volume 145: Topics in Stochastic Analysis and Nonparametric Estimation

Editors: Pao-Liu Chow, Boris Mordukhovich, and George Yin

Volume 146: Algorithms in Algebraic Geometry

Editors: Alicia Dickenstein, Frank-Olaf Schreyer, and Andrew J.
Sommese

Volume 147: Symmetric Functionals on Random Matrices and Random Matchings Problems by Grzegorz A. Rempała and Jacek Wesołowski

Volume 148: Software for Algebraic Geometry

Editors: Michael E. Stillman, Nobuki Takayama, and Jan Verschelde

Volume 149: Emerging Applications of Algebraic Geometry

Editors: Mihai Putinar and Seth Sullivant

Volume 150: Mathematics of DNA Structure, Function, and Interactions

Editors: Craig John Benham, Stephen Harvey, Wilma K. Olson,
De Witt L. Sumners, and David Swigon

Volume 151: Nonlinear Computational Geometry

Editors: Ioannis Z. Emiris, Frank Sottile, and Thorsten Theobald

Volume 152: Towards Higher Categories

Editors: John C. Baez and J. Peter May

Volume 153: Nonlinear Conservation Laws and Applications

Editors: Alberto Bressan, Gui-Qiang Chen, Marta Lewicka, and
Dehua Wang

Volume 154: Mixed Integer Nonlinear Programming

Editors: Jon Lee and Sven Leyffer