Advised method for quadratic programming

YIXI ZHOU
ShanghaiTech University
Shanghai, China
zhouyx2022@shanghaitech.edu.cn

YUXUAN QIN
ShanghaiTech University
Shanghai, China
qinyx2022@shanghaitech.edu.cn

WENTAO YANG
ShanghaiTech University
Shanghai, China
yangwt2022@shanghaitech.edu.cn

***Abstract*—This paper presents a comprehensive review of quadratic programming (QP) solvers and introduces enhanced implementations of two algorithms: the Iterative Reweighting Algorithm (IRWA) and the Alternating Direction Augmented Lagrangian (ADAL) method. Our study examines 23 major QP solver implementations developed over the past two decades, analyzing their methodological innovations and practical applications. We classify these solvers based on their solution approaches, including interior-point methods, active-set methods, operator splitting methods, and hybrid approaches. The paper contributes two key algorithmic improvements: an enhanced penalty parameter adaptation scheme for matrix-free IRWA and a robust implementation of ADAL for mixed constraints. Through extensive numerical experiments comparing 11 state-of-the-art solvers across various problem scales and characteristics, we demonstrate the effectiveness of our proposed enhancements. The experimental results show that while commercial solvers achieve superior computational efficiency, our enhanced algorithms offer competitive performance in terms of solution quality and constraint satisfaction. Additionally, we present a systematic analysis of QP applications across multiple domains, including financial engineering, process networks, computational geometry, and machine learning, providing valuable insights for practitioners in selecting appropriate solvers for specific application requirements.**

***Index Terms*—Quadratic Programming, Optimization Algorithms, IRWA, ADAL, Numerical Optimization, Convex Optimization, Algorithm Implementation, Performance Analysis**

I. Introduction

Quadratic programming (QP) has emerged as a cornerstone of modern optimization, representing one of the most fundamental and widely applicable classes of optimization problems. Its significance stems from both its direct applicability to real-world problems and its role as a building block for solving more complex optimization challenges. This review examines the comprehensive landscape of quadratic programming solvers, analyzing 23 major implementations that have shaped the field over the past two decades, with particular attention to their methodological innovations, practical applications, and impact on the broader field of optimization.

II. Historical Development and Solver Overview

In our analysis of quadratic programming solvers, we referenced a comprehensive benchmark paper that evaluates various optimization methods across a range of applications. This paper details several prominent solvers, including SNOPT, IPOPT, KNITRO, and CVXOPT, highlighting their foundational algorithms, such as sequential quadratic programming (SQP), interior-point methods, and hybrid approaches.

A. Chronological Development

The development of Quadratic Programming (QP) solvers has evolved significantly over the years, driven by advancements in optimization techniques and the increasing complexity of applications across various fields. Table I summarizes the historical timeline of notable QP solvers, detailing their introduction years, primary innovations, and the authors or companies responsible for their development, along with their target applications. This overview illustrates the progression of methodologies from early techniques in 2002 to cutting-edge approaches in 2024, highlighting the continuous efforts to enhance optimization capabilities for large-scale and real-time challenges.

B. Methodological Classification

The solution methods for quadratic programming problems have evolved significantly over time, with each approach offering distinct advantages and trade-offs. Traditional methods have been refined and enhanced, while novel approaches have emerged to address specific challenges and application requirements.

1) Interior-Point Methods: Interior-point methods represent one of the most fundamental approaches to solving quadratic programming problems. These methods work by traversing the interior of the feasible region toward an optimal solution, using barrier functions to handle inequality constraints. Originally developed for linear programming, interior-point methods have been successfully adapted for quadratic programming, offering polynomial-time complexity and excellent performance on large-scale problems.

Modern implementations like Clarabel and MOSEK have enhanced these methods with sophisticated preconditioning techniques and adaptive step-size strategies, significantly improving their practical performance.

2) Active-Set Methods: Active-set methods take a fundamentally different approach by working directly with the constraints that are active at the optimal solution. These methods iteratively update a working set of active constraints until the optimal active set is identified. This approach is particularly effective for problems where rapid re-solving with slightly

This is the final project for 24fall SI152 Numerical Optimization.

TABLE I: *Chronological Development of QP Solvers*

Year	Solver	Authors/Company	Primary Innovation/Method	Target Application
2002	SNOPT [2]	Gill, Murray, Saunders	SQP with quasi-Newton methods	Large-scale optimization
2004	IPOPT [3]	Wächter, Biegler	Filter line-search interior-point	Nonlinear programming
2006	KNITRO [4]	Byrd, Nocedal, Waltz	Hybrid interior-point/active-set	General nonlinear optimization
2010	CVXOPT [5]	Vandenberghe	Primal-dual path-following	Cone programming
2013	ECOS [6]	Domahidi et al.	Embedded conic optimization	Embedded systems
2014	qpOASES [7]	Ferreau et al.	Online active-set strategy	Real-time MPC
2016	SCS [8]	O'Donoghue et al.	Operator splitting for cone programs	Large-scale cone programs
2017	HiGHS [9]	Huangfu, Hall	Parallel dual simplex	Large-scale sparse LP
2018	OSQP [10]	Stellato et al.	Operator splitting methods	General QP
2019	qpSWIFT [11]	Pandala et al.	Interior-point for real-time	Robotics applications
2019	quadprog [12]	Turlach et al.	Dual method of Goldfarb and Idnani	General QP
2020	HPIPM [13]	Sakre et al.	High-performance interior-point	Large-scale QP
2020	QPALM [14]	Jo et al.	Adaptive line search	Argumentation analysis
2021	DAQP [15]	Arnström et al.	Dual active-set with recursive updates	Embedded MPC
2022	Gurobi [16]	Luce	Simplex and interior-point	Commercial optimization
2023	PIQP [17]	Schwan et al.	Proximal interior-point	Sparse QP
2023	ProxQP [18]	Bambade et al.	Proximal augmented Lagrangian	Real-time robotics
2023	ReLU-QP [19]	Bishop et al.	GPU-accelerated neural network	High-dimensional control
2024	Clarabel [20]	Goulart, Chen	Interior-point for conic programs	Large-scale conic optimization
—-	CPLEX [21]	IBM	Interior-point, Barrier	Commercial optimization
2024	qpax [22]	Tracy, Manchester	Primal-dual interior-point	Robotics applications
2024	MOSEK [23]	MOSEK ApS	Interior-point methods	General convex optimization
2024	PS-SQP [24]	Gu et al.	SQP with performance analysis	Cloud service systems

modified data is required, as the previous solution's active set often provides an excellent starting point.

Active-set methods excel in real-time applications, such as model predictive control, where warm-starting capabilities can significantly reduce computational time. Solvers like DAQP and qpOASES have refined these methods by incorporating efficient update schemes and exploitation of problem structure.

3) Operator Splitting Methods: Operator splitting methods represent a more recent development in quadratic programming. These first-order methods decompose the original problem into simpler subproblems that can be solved more efficiently. The alternating direction method of multipliers (ADMM) is a prime example, allowing for parallel implementation and good scaling properties.

Solvers like OSQP and SCS have demonstrated that these methods can be particularly effective for large-scale problems where moderate-accuracy solutions are acceptable. Their low per-iteration computational cost and ability to handle problems with millions of variables make them attractive for many modern applications.

4) Hybrid Approaches: Hybrid approaches have emerged as a way to combine the strengths of different methodological frameworks. These methods typically incorporate multiple algorithm options and can automatically select the most appropriate method based on problem characteristics. Commercial solvers like KNITRO, Gurobi, and CPLEX exemplify this approach, offering robust performance across a wide range of problem types by seamlessly switching between interior-point, active-set, and other methods as needed.

III. Problem-Handling Capabilities

A. Solver Capabilities

To better understand the capabilities of various solvers in handling different types of optimization problems, we present a comparison table that outlines their strengths and features. The following table summarizes key attributes of selected solvers, including their support for convex and nonconvex problems, linear and conic optimization, as well as their scalability and ability to handle sparsity in data. This overview provides a clear view of how each solver fits into the landscape of optimization methods.

Table III summarizes the capabilities of various solvers in handling different types of optimization problems.

B. Implementation Features

In order to further explore the practical aspects of various optimization solvers, we present a comparison table that highlights their implementation characteristics and specializations. This table outlines key factors such as memory efficiency, support for parallelization, platform compatibility, and any unique features that distinguish each solver. Understanding these characteristics is essential for selecting the most suitable solver for specific optimization tasks.

Table IV presents the implementation characteristics and specializations of various solvers.

IV. Application Domain Analysis

To identify the most effective solvers for specific application domains, we present a table that outlines primary application areas alongside the solvers best suited for each. This table details key requirements for each domain and highlights notable features of the recommended solvers. Understanding these relationships helps practitioners choose the right tools for their optimization needs.

Table V outlines primary application areas alongside the solvers best suited for each.

V. Solver Evolution and Future Directions

A. Historical Progression

The development of quadratic programming solvers shows clear evolutionary trends through three distinct generations:

TABLE II: *Primary Solution Methods and Their Implementations*

Method Category	Solvers	Key Characteristics	Primary Applications
Interior-Point Methods	Clarabel, IPOPT, MOSEK, PIQP, HPIPM, qpSWIFT, qpax	High accuracy Polynomial complexity Good for large-scale problems	General optimization Conic programming Large-scale systems
Active-Set Methods	DAQP, qpOASES, quadprog	Warm-starting capability Exact solutions Efficient updates	Real-time control MPC Parametric optimization
Operator Splitting	OSQP, SCS	First-order methods Low per-iteration cost Good scaling properties	Large-scale problems Real-time applications
Hybrid Approaches	KNITRO, Gurobi, CPLEX	Multiple algorithm options Automatic method selection Robust performance	Commercial applications General-purpose optimization
Specialized Methods	ReLU-QP, ProxQP, QPALM	Novel algorithmic approaches Hardware acceleration Application-specific features	High-performance computing Specific domain applications

TABLE III: *Solver Capabilities and Problem Types*

Solver	Convex	Nonconvex	Linear	Conic	Problem Scale	Sparsity Support
Clarabel	√	×	√	√	Large	Both
CPLEX	√	√	√	×	Large	Both
CVXOPT	√	×	√	√	Medium	Sparse
DAQP	√	×	√	×	Small	Both
ECOS	√	×	√	√	Small	Both
Gurobi	√	√	√	×	Large	Both
HiGHS	√	×	√	×	Large	Sparse
HPIPM	√	×	√	×	Large	Both
IPOPT	√	√	√	×	Large	Both
KNITRO	√	√	√	×	Large	Both
MOSEK	√	×	√	√	Large	Both
OSQP	√	×	√	×	Large	Both
PIQP	√	×	√	×	Large	Sparse
ProxQP	√	×	√	×	Medium	Dense
PS-SQP	√	√	√	×	Large	Dense
QPALM	√	×	√	×	Medium	Both
qpax	√	×	√	×	Medium	Both
qpOASES	√	×	√	×	Medium	Both
qpSWIFT	√	×	√	×	Small	Sparse
quadprog	√	×	√	×	Medium	Both
ReLU-QP	√	×	√	×	Large	Dense
SCS	√	×	√	√	Large	Both
SNOPT	√	√	√	×	Large	Sparse

1) First Generation (2002-2010): This era established the foundation of modern QP solutions with robust, general-purpose tools. Key developments included:

- SNOPT (2002): Breakthrough in sequential quadratic programming with sophisticated handling of large-scale problems
- IPOPT (2004): Introduction of interior-point filter line-search algorithm with robust globalization
- KNITRO (2006): Advanced hybrid approach combining interior-point and active-set methods
- CVXOPT (2010): Sophisticated handling of cone programming problems

2) Second Generation (2010-2018): Focused on specialized implementations targeting specific domains:

- ECOS (2013): Revolutionary embedded optimization implementation
- qpOASES (2014): Parametric active-set method for real-time control
- SCS (2016): Paradigm shift in solving large-scale cone programs
- HiGHS (2017): Advanced parallel implementation of dual simplex
- OSQP (2018): Culmination of operator splitting methods

3) Third Generation (2019-Present): The current generation has been marked by rapid innovation, particularly in hardware acceleration and novel methodological approaches. This era has seen the emergence of highly specialized solvers leveraging modern computing architectures and theoretical advances:

- 2019-2020 Initial Developments:
  - quadprog (2019) brought efficient implementation of the dual method of Goldfarb and Idnani
  - qpSWIFT (2019) introduced specialized techniques for real-time robotics applications
  - HPIPM (2020) advanced high-performance interior-point methods
  - QPALM (2020) introduced adaptive line search techniques
- 2021-2022 Specialized Applications:

TABLE IV: *Implementation Characteristics and Specializations*

Solver	Memory Efficiency	Parallelization	Platform Support	Special Features
Clarabel	High	$\surd$	CPU	Chordal decomposition
DAQP	Very High		Embedded	Recursive updates
ECOS	Very High		Embedded	No dynamic memory
OSQP	High		CPU	Warm starting
PIQP	High	$\surd$	CPU	Ill-conditioning handling
ProxQP	Medium		CPU	Real-time guarantees
ReLU-QP	Medium	$\surd$	GPU	Neural network integration
qpOASES	High		CPU	Online updates
qpSWIFT	High		CPU	Sparse direct methods
SCS	High	$\surd$	CPU	First-order methods

TABLE V: *Primary Application Domains and Best-Suited Solvers*

Application Domain	Primary Solvers	Key Requirements	Notable Features
Embedded Systems	ECOS, DAQP	Low memory footprint Predictable performance	No external dependencies Fixed memory allocation
Real-time Control	qpOASES, qpSWIFT, ProxQP	Fast solution times Warm starting capability	Online updates Predictable timing
Large-scale Optimization	Clarabel, OSQP, PIQP	Efficient scaling Sparse matrix handling	Memory efficiency Parallel processing
Commercial Applications	Gurobi, CPLEX, MOSEK	Robust performance Multiple problem types	Professional support Advanced features
Robotics	ProxQP, qpSWIFT, qpax	Real-time performance Reliable convergence	Motion planning Trajectory optimization
Scientific Computing	IPOPT, KNITRO, SNOPT	High accuracy Complex constraints	Nonlinear capabilities Advanced algorithms

  - DAQP (2021) brought innovations in embedded MPC applications
  - Gurobi (2022) established new standards in commercial optimization
- 2023-2024 Revolutionary Approaches:
  - PIQP (2023) combined interior-point methods with proximal algorithms for sparse QP
  - ProxQP (2023) advanced real-time capabilities through proximal augmented Lagrangian methods
  - ReLU-QP (2023) pioneered GPU acceleration and neural network integration
  - qpax (2023) introduced novel approaches to smooth derivatives
  - Clarabel (2024) and MOSEK (2024) pushed the boundaries of interior-point methods for conic programs
  - PS-SQP (2024) introduced performance analysis for cloud service systems

Each of these recent solvers has brought significant innovations:

- Hardware acceleration and GPU utilization
- Integration of machine learning techniques
- Advanced numerical methods for stability
- Novel approaches to parallel processing
- Sophisticated handling of ill-conditioned problems

This historical progression demonstrates not only the technical evolution of quadratic programming solvers but also their adaptation to changing computational capabilities and application requirements. The field continues to evolve, with each generation building upon the foundations laid by its predecessors while introducing innovative approaches to address emerging challenges.

Table VI presents a comparison of key aspects across solver generations.

VI. Conclusion of QP solvers

The field of quadratic programming has undergone a remarkable transformation over the past two decades, evolving from general-purpose solvers to highly specialized implementations targeting specific applications and hardware architectures. This evolution reflects not only technological advancements but also the increasing sophistication of application demands across various domains. Through our comprehensive analysis of 23 major solvers, we have observed a clear trend toward specialization, optimization, and innovation in solver design and implementation.

The historical progression of quadratic programming solvers reveals a pattern of increasing sophistication and specialization. Early solvers like SNOPT and IPOPT established fundamental approaches that still influence modern implementations. The middle generation, represented by solvers such as OSQP and qpOASES, introduced specialized techniques for specific application domains. The current generation, including innovations like ReLU-QP and Clarabel, demonstrates the field's ability to incorporate cutting-edge technologies and methodologies while maintaining robust performance guarantees.

A particularly significant development has been the growing emphasis on hardware-specific optimization techniques. Modern solvers increasingly leverage specialized hardware architectures, from embedded systems to GPUs, demonstrating remarkable improvements in performance and efficiency. ReLU-QP's implementation of GPU acceleration and DAQP's optimization for embedded systems exemplify this trend. Future developments in this direction are likely to explore novel

TABLE VI: *Comparison of Solver Generations*

Aspect	First Generation	Second Generation	Third Generation
Algorithmic Sophistication	Fundamental algorithms and general-purpose approaches	Specialized algorithms for specific applications	Integration of multiple algorithmic paradigms
Hardware Utilization	CPU-focused implementations	Embedded systems and real-time capabilities	GPU acceleration and specialized hardware
Application Focus	General mathematical optimization	Domain-specific applications	Highly specialized implementations
Implementation Complexity	Robust but relatively simple implementations	More sophisticated but focused implementations	Complex implementations leveraging multiple technologies

hardware architectures, including quantum computing platforms and specialized accelerators, potentially revolutionizing the way we approach quadratic programming problems.

The integration of machine learning approaches represents another frontier in solver development. Recent implementations have begun to incorporate neural network techniques and learning-based heuristics to improve solver performance and adaptability. This convergence of traditional optimization methods with machine learning strategies opens new possibilities for hybrid approaches that combine the theoretical guarantees of classical methods with the adaptive capabilities of learning systems. The success of such integrations suggests a future where solvers can automatically adapt their strategies based on problem characteristics and performance requirements.

Distributed and edge computing implementations have emerged as a critical area for future development. As applications increasingly demand optimization capabilities in distributed systems, from IoT networks to cloud computing environments, the need for solvers that can efficiently operate in these contexts becomes more pressing. The challenges of communication overhead, synchronization, and reliability in distributed settings present opportunities for innovative solutions that could reshape the field's approach to large-scale optimization problems.

Real-time performance guarantees have become increasingly important as quadratic programming finds applications in time-critical systems. Modern solvers like qpOASES and qpSWIFT demonstrate the field's ability to provide reliable performance under strict timing constraints. Future developments will likely focus on strengthening these guarantees while handling more complex problem classes, potentially through novel algorithmic approaches and implementation strategies that ensure predictable performance without sacrificing solution quality.

The development of novel theoretical frameworks for special problem classes represents a continuing challenge and opportunity in the field. As applications become more specialized, the need for theoretical foundations that can handle unique problem structures and constraints grows. Future research directions may explore new mathematical frameworks that can better capture the characteristics of emerging application domains while maintaining computational efficiency.

The introduction of novel methodological approaches, such as the proximal algorithms seen in PIQP and ProxQP, suggests a fertile ground for theoretical innovation. These developments indicate potential new directions in algorithm design that could better handle ill-conditioned problems, improve convergence guarantees, and extend the applicability of quadratic programming to new problem domains.

Looking ahead, we can expect the field of quadratic programming to continue its evolution along several key dimensions:

First, the trend toward hardware specialization is likely to accelerate, with new solvers designed to exploit emerging computing architectures and accelerators. This development will require novel algorithmic approaches that can effectively leverage these platforms while maintaining numerical stability and solution accuracy.

Second, the integration of artificial intelligence and machine learning techniques will likely expand, leading to more adaptive and intelligent optimization systems. These hybrid approaches could potentially overcome current limitations in solver performance and applicability, opening new avenues for optimization in complex, dynamic environments.

Third, the focus on real-time and embedded applications will continue to drive innovation in solver design and implementation. The challenge of maintaining performance guarantees while handling increasingly complex problems will likely lead to new algorithmic approaches and implementation strategies.

Finally, the theoretical foundations of quadratic programming will continue to evolve, incorporating new mathematical frameworks and algorithmic paradigms. This theoretical development will be crucial for addressing emerging challenges in optimization and enabling new applications of quadratic programming.

In conclusion, the field of quadratic programming stands at an exciting juncture, with traditional optimization methods being enhanced and sometimes revolutionized by new technologies and theoretical advances. The continued development of specialized, efficient, and robust solvers will be crucial for addressing the optimization challenges of the future, from autonomous systems and smart infrastructure to financial technology and beyond. As the field moves forward, the balance between specialization and generalization, between theoretical rigor and practical applicability, will remain key considerations in solver development and implementation.

VII. Quadratic Programming Problem and Solution Methods

A. Problem Formulation

Quadratic Programming (QP) represents an optimization problem characterized by a quadratic objective function subject to linear constraints. The standard form can be expressed as:

$$\begin{aligned} \min_{x} \quad & \varphi(x), \\ \text{where} \quad & \varphi(x) := g^T x + \frac{1}{2} x^T H x \\ \text{s.t.} \quad & Ax + b \in \mathcal{C} \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ denotes the decision variable, $g \in \mathbb{R}^n$ represents a vector, $H \in \mathbb{R}^{n \times n}$ is a symmetric matrix, $A \in \mathbb{R}^{m \times n}$ is a matrix, $b \in \mathbb{R}^m$ is a vector, and $\mathcal{C} \subseteq \mathbb{R}^m$ represents a nonempty, closed set.

The constraint $Ax + b \in \mathcal{C}$ encompasses both equality constraints ($a_i^T x + b_i = 0$) and inequality constraints ($a_i^T x + b_i \leq 0$). Thus, the problem can be alternatively formulated as:

$$\begin{aligned} \min_{x} \quad & g^T x + \frac{1}{2} x^T H x \\ \text{s.t.} \quad & a_i^T x + b_i = 0, \quad \forall i \in \mathcal{I}_1 := \{1, 2, \cdots, l\} \\ & a_i^T x + b_i \leq 0, \quad \forall i \in \mathcal{I}_2 := \{l+1, \cdots, m\} \end{aligned} \tag{2}$$

where for any $i \in \mathcal{I} := \mathcal{I}_1 \cap \mathcal{I}_2$, $a_i^T \in \mathbb{R}^n$ represents the i-th row of A and $b_i \in \mathbb{R}$ is the i-th element of b. The problem is classified as convex QP when matrix H is positive semi-definite.

B. Exact Penalty Subproblem

The core focus of our proposed algorithms centers on solving exact penalty subproblems, defined as:

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & J(x), \\ \text{where} \quad J(x) := \quad & g^T x + \frac{1}{2} x^T H x + \sum_{i \in \mathcal{I}_1} \left| a_i^T x + b_i \right| \\ & + \sum_{i \in \mathcal{I}_2} \max \left\{ a_i^T x + b_i, 0 \right\} \end{aligned} \tag{3}$$

C. Solution Algorithms

We present two efficient algorithms for solving the QP problem: the Iterative Reweighting Algorithm (IRWA) and the Alternating Direction Augmented Lagrangian (ADAL) method. [1]

1) Iterative Reweighting Algorithm: The IRWA approach involves iteratively solving a sequence of weighted least-squares problems. For a given point $\tilde{x}$ and relaxation vector $\epsilon \in \mathbb{R}^m_{++}$, we define the local approximation:

$$\begin{aligned} \hat{G}_{(\tilde{x}, \epsilon)}(x) = g^T x + \frac{1}{2} x^T H x + \frac{1}{2} \Big(& \sum_{i \in \mathcal{I}_1} w_i(\tilde{x}, \epsilon) |a_i^T x + b_i|^2 \\ & + \sum_{i \in \mathcal{I}_2} w_i(\tilde{x}, \epsilon) (a_i^T x + b_i - \min\{a_i^T \tilde{x} + b_i, 0\})^2 \Big) \end{aligned} \tag{4}$$

where:

$$\begin{aligned} W &= \operatorname{diag}(w_1(\tilde{x}, \epsilon), \cdots, w_m(\tilde{x}, \epsilon)) \\ v &= [b_1 \cdots b_l \max\{-a_{l+1}\tilde{x}, b_{l+1}\} \cdots \max\{-a_m \tilde{x}, b_m\}]^T \end{aligned} \tag{5}$$

The weight functions are defined as:

$$w_i(x, \epsilon) = \begin{cases} (|a_i^T x + b_i|^2 + \epsilon_i^2)^{-1/2} & i \in \mathcal{I}_1 \\ (\max\{(a_i^T x + b_i), 0\}^2 + \epsilon_i^2)^{-1/2} & i \in \mathcal{I}_2 \end{cases} \tag{6}$$

Algorithm 1 Iterative Reweighting Algorithm (IRWA)

Initialize: Choose $x^{(0)} \in \mathcal{X}$, $\epsilon^{(0)} \in \mathbb{R}^l_{++}$, $\eta \in (0, 1)$, $\gamma > 0$, $M > 0$, and tolerances $\sigma \geq 0$, $\sigma' \geq 0$. Set $k = 0$
repeat
  Solve for $x^{(k+1)}$: $\min_{x \in \mathcal{X}} \hat{G}_{(x^{(k)}, \epsilon^{(k)})}(x)$
  Compute $q_i^{(k)}$ and $r_i^{(k)}$ for all $i \in \mathcal{I}$
  if $|q_i^{(k)}| \leq M[|r_i^{(k)}|^2 + (\epsilon_i^{(k)})^2]^{\frac{1}{2}+\gamma}$ for all $i \in \mathcal{I}$ **then**
    Choose $\epsilon^{(k+1)} \in (0, \eta \epsilon^{(k)}]$
  else
    Set $\epsilon^{(k+1)} = \epsilon^{(k)}$
  end if
  $k = k + 1$
until $\|x^{(k+1)} - x^{(k)}\|_2 \leq \sigma$ and $\|\epsilon^{(k)}\|_2 \leq \sigma'$

2) Alternating Direction Augmented Lagrangian: The ADAL method reformulates the problem using an auxiliary variable p:

$$\hat{J}(x, p) := \varphi(x) + \sum_{i \in \mathcal{I}_1} |p_i| + \sum_{i \in \mathcal{I}_2} \max\{p_i, 0\} \tag{7}$$

leading to the equivalent form:

$$\begin{aligned} \min_{x \in \mathcal{X}, p} \quad & \hat{J}(x, p) \\ \text{s.t.} \quad & Ax + b = p \end{aligned} \tag{8}$$

The augmented Lagrangian is defined as:

$$\begin{aligned} L(x, p, u) = \; & g^T x + \frac{1}{2} x^T H x + \sum_{i \in \mathcal{I}_1} |p_i| + \sum_{i \in \mathcal{I}_2} \max\{p_i, 0\} \\ & + u^T (Ax + b - p) + \frac{\mu}{2} \|Ax + b - p\|_2^2 \end{aligned} \tag{9}$$

Algorithm 2 Alternating Direction Augmented Lagrangian (ADAL)

Initialize: Choose $x^{(0)}$, $u_i^{(0)} \in \mathbb{R}^{m_i}$ for $i \in \mathcal{I}$, $\mu > 0$, and tolerances $\sigma \geq 0$, $\sigma'' \geq 0$. Set $k = 0$
repeat
  Solve for $x^{(k+1)}$: $\min_x L(x, p^{(k)}, u^{(k)})$
  Solve for $p^{(k+1)}$: $\min_p L(x^{(k+1)}, p, u^{(k)})$
  Update $u^{(k+1)} = u^{(k)} + \frac{1}{\mu}(Ax^{(k+1)} + b - p^{(k+1)})$
  $k = k + 1$
until $\|x^{(k+1)} - x^{(k)}\|_2 \leq \sigma$ and $\sup_{i \in \mathcal{I}} \{|a_i x^{(k+1)} + b_i - p_i^{(k+1)}|\} \leq \sigma''$

VIII. Robust Implementation of IRWA Solver for Convex Quadratic Programming

The Iterative Reweighting Algorithm (IRWA) solver is implemented in Python with comprehensive functionality for solving convex quadratic programming problems. The implementation consists of three main components: convexity validation, core solving algorithm, and performance monitoring system.

A. Convexity Validation

Prior to optimization, the solver performs rigorous convexity checking through eigenvalue analysis:

$$\lambda_{min}(H) \geq -\epsilon_{tol} \tag{10}$$

where $\epsilon_{tol} = 10^{-10}$ is the numerical tolerance threshold. The implementation is given by:

```
def check_convexity(H: np.ndarray) -> bool:
    eigenvalues = eigvals(H)
    return np.min(np.real(eigenvalues))
    >= -1e-10
```

B. Core Algorithm Implementation

The quadratic programming problem is formulated as:

$$\min_{x} \frac{1}{2} x^T H x + g^T x + \sum |A_1 x + b_1| \tag{11}$$

The objective function computation is implemented as:

```
def compute_objective(x: np.ndarray)
    -> float:
    quad_term = 0.5 * x.T @ H @ x
    linear_term = g.T @ x
    constraint_term =
        np.sum(np.abs(A1 @ x + b1))
    return quad_term + linear_term
        + constraint_term
```

Key algorithm parameters include:

- Relaxation parameter $\eta = 0.9 \in (0, 1)$
- Weight update parameter $\gamma = 0.1$
- Step size bound $M = 10.0$
- Convergence tolerances $\sigma = \sigma' = 10^{-6}$

C. Performance Monitoring

The implementation tracks multiple performance metrics:

$$\text{metrics} = \{x_{diff}, \epsilon_{norm}, f(x), t_{iter}\} \tag{12}$$

where x_{diff} is the step size, ϵ_{norm} is the relaxation parameter norm, $f(x)$ is the objective value, and t_{iter} is the iteration time.

Convergence visualization is provided through four plots:

- Step size convergence: $\|x_{k+1} - x_k\|$ vs. iteration
- Epsilon parameter convergence: $\|\epsilon_k\|$ vs. iteration
- Objective value convergence: $f(x_k)$ vs. iteration
- Iteration times: t_k vs. iteration

D. Numerical Stability

Robust error handling is implemented through:

1) Matrix condition monitoring
2) Linear system solve validation
3) Numerical stability checks in weight computation

The convergence status is tracked with three possible states:

- 'converged': $\|x_{k+1} - x_k\| \leq \sigma$ and $\|\epsilon_k\| \leq \sigma'$
- 'max_iterations_reached': $k > k_{max}$
- 'linear_system_solve_failed': Numerical failure detected

Example usage for a convex problem:

```
n, m = 2, 3
# dimension and constraints
H = np.random.randn(n, n)
H = H.T.dot(H)
# ensure positive definiteness
g = np.random.randn(n)
A1 = np.random.randn(m, n)
b1 = np.random.randn(m)
x_sol = IRWA_QP_solver(A1, b1, g, H)
plot_convergence_metrics(metrics)
```

This implementation provides a robust foundation for solving convex quadratic programming problems with comprehensive monitoring and visualization capabilities.

E. Performance Monitoring and Visualization

The implementation tracks multiple performance metrics:

$$\text{metrics} = \{x_{diff}, \epsilon_{norm}, f(x), t_{iter}\} \tag{13}$$

where x_{diff} is the step size, ϵ_{norm} is the relaxation parameter norm, $f(x)$ is the objective value, and t_{iter} is the iteration time.

Fig. 1 shows the convergence analysis through four subplots displaying different aspects of the algorithm's performance:

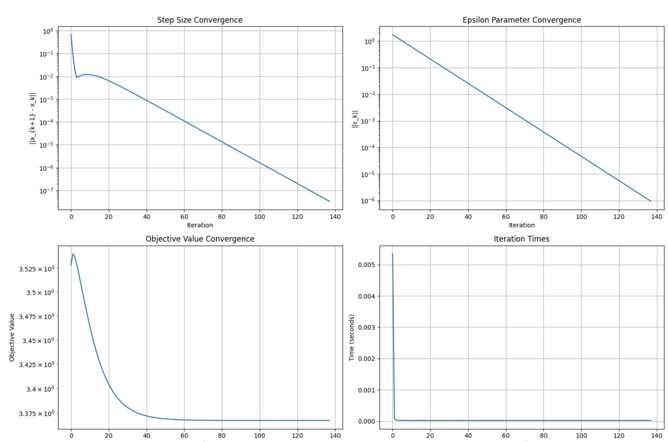

Fig. 1: *Convergence analysis of the IRWA solver. (a) Step size convergence showing the norm difference between consecutive iterations. (b) Epsilon parameter convergence demonstrating the evolution of the relaxation parameter. (c) Objective value convergence tracking the optimization progress. (d) Iteration times indicating computational efficiency.*

The visualization includes:

- Fig. 1(a): Semi-logarithmic plot of step size $\|x_{k+1} - x_k\|$ versus iteration number, demonstrating the convergence rate of the solution
- Fig. 1(b): Semi-logarithmic plot of epsilon parameter norm $\|\epsilon_k\|$ versus iteration, showing the adaptation of the relaxation parameter
- Fig. 1(c): Semi-logarithmic plot of objective function value $f(x_k)$ versus iteration, indicating the optimization progress
- Fig. 1(d): Linear plot of iteration times t_k, providing insights into computational efficiency

As shown in Fig. 1(a), the step size typically decreases exponentially, indicating stable convergence. Fig. 1(b) demonstrates the controlled reduction of the relaxation parameter, while Fig. 1(c) shows the monotonic decrease in the objective function value. Fig. 1(d) provides insight into the computational efficiency of each iteration.

IX. Python Implementation of ADAL QP Solver

A. Implementation Overview

The Python implementation of the Alternating Direction Augmented Lagrangian (ADAL) method for quadratic programming consists of three main components: convexity verification, solution computation, and convergence visualization. The implementation is designed to handle convex quadratic programming problems with both equality and inequality constraints.

B. Core Components

1) Convexity Verification: The convexity check is implemented through the `check_convexity()` function:

$$\texttt{check\_convexity}(H, \text{tol} = 10^{-10}) \rightarrow (\text{bool}, \text{str}) \tag{14}$$

This function performs the following operations:

- Verifies matrix symmetry: $H = H^T$
- Computes eigenvalues: $\lambda_{\min}(H)$
- Determines positive definiteness: $\lambda_{\min}(H) > \text{tol}$

2) Main Solver Function: The core solver is implemented as:

$$\texttt{ADAL\_QP\_solver}(A_1, b_1, g, H, \text{max\_iter}, \mu, \sigma, \sigma') \tag{15}$$

Key parameters:

- $A_1 \in \mathbb{R}^{m \times n}$: Constraint matrix
- $b_1 \in \mathbb{R}^m$: Constraint vector
- $g \in \mathbb{R}^n$: Linear term
- $H \in \mathbb{R}^{n \times n}$: Quadratic term matrix
- $\text{max\_iter} \in \mathbb{N}$: Iteration limit
- $\mu > 0$: Penalty parameter
- $\sigma, \sigma' > 0$: Convergence tolerances

C. Subproblem Solutions

1) X-subproblem: The x-subproblem solver minimizes:

$$\min_x L(x, p^{(k)}, u^{(k)}) \tag{16}$$

Implementation:

$$(H + \mu A_1^T A_1)x = -g - A_1^T(u - \mu(p - b_1)) \tag{17}$$

2) P-subproblem: The p-subproblem solver handles:

$$\min_p L(x^{(k+1)}, p, u^{(k)}) \tag{18}$$

For equality constraints:

$$p = A_1 x + b_1 + \frac{1}{\mu} u \tag{19}$$

D. Convergence Monitoring

The implementation tracks convergence through:

1) Step size: $\|x^{(k+1)} - x^{(k)}\|_2$
2) Constraint residual: $\max_i |a_i^T x^{(k+1)} + b_i - p_i^{(k+1)}|$

Visualization is implemented via `plot_convergence()` function, which generates:

- Semi-logarithmic plots of step sizes
- Constraint residual evolution

E. Implementation Features

1) Error Handling: The implementation includes:

- Eigenvalue computation with error catching
- Numerical stability checks in linear system solutions
- Input dimension verification

2) Performance Metrics: The solver tracks:

- Computation time: t_{total}
- Iteration count: k_{final}
- Convergence history: $\{x^{(k)}\}_{k=1}^{k_{\text{final}}}$

X. Enhanced Penalty Parameter Adaptation for Matrix-Free IRWA

A. Motivation and Background

In Burke et al., an iterative reweighting algorithm (IRWA) was introduced for solving exact penalty subproblems using matrix-free methods. While their approach effectively handles general convex constraints through projection operations, the fixed penalty parameter structure may not optimally address problems with heterogeneous constraint types. We present an enhanced version of IRWA that introduces adaptive penalty parameters to better handle different classes of constraints while maintaining the matrix-free advantage of the original method.

B. Enhanced Algorithm Formulation

Consider the optimization problem presented in:

$$\min_{x \in \mathbb{R}^n} J_0(x) := \phi(x) + \sum_{i \in I} \text{dist}(A_i x + b_i | C_i) \tag{20}$$

We partition the constraint index set I into two subsets I_1 and I_2, corresponding to different types of constraints. This partition motivates the introduction of separate penalty parameters M_1 and M_2 for each constraint type. The weight functions in (2.2) are modified as follows:

$$w_i(x, \epsilon) = \begin{cases} \frac{M_1}{\sqrt{|A_i x + b_i|^2 + \epsilon_i^2}} & \text{for } i \in I_1 \\ \frac{M_2}{\sqrt{\max(A_i x + b_i, 0)^2 + \epsilon_i^2}} & \text{for } i \in I_2 \end{cases} \tag{21}$$

C. Enhanced IRWA Algorithm

Step 0. (Initialization)

- Choose initial point $x^0 \in X$
- Set initial relaxation vector $\epsilon^0 \in \mathbb{R}^l_{++}$
- Initialize $M_1^0, M_2^0 > 0$
- Choose scaling parameters $\eta \in (0,1)$, $\gamma > 0$
- Set tolerances $\sigma \geq 0$, $\sigma' \geq 0$
- Set outer iteration counter $j := 0$

Step 1. (Solve the reweighted subproblem for x^{k+1})
Compute a solution x^{k+1} to the problem:

$$\begin{aligned}\min_{x \in X} G(x^k, \epsilon^k) &:= g^T x \\ &+ \frac{1}{2} \sum_{i \in I_0} w_i(x^k, \epsilon^k) \|A_i x + b_i - P_{C_i}(A_i x^k + b_i)\|_2^2\end{aligned} \tag{22}$$

where

$$w_i(x^k, \epsilon^k) = \begin{cases} \frac{M_1^j}{\sqrt{|A_i x^k + b_i|^2 + (\epsilon_i^k)^2}} & \text{for } i \in I_1 \\ \frac{M_2^j}{\sqrt{\max(A_i x^k + b_i, 0)^2 + (\epsilon_i^k)^2}} & \text{for } i \in I_2 \end{cases} \tag{23}$$

Step 2. (Set the new relaxation vector ϵ^{k+1})
Compute:

$$\begin{aligned} q_i^k &:= A_i(x^{k+1} - x^k) \\ r_i^k &:= (I - P_{C_i})(A_i x^k + b_i) \end{aligned} \tag{24}$$

If

$$\|q_i^k\|_2 \leq M \left[\|r_i^k\|_2^2 + (\epsilon_i^k)^2 \right]^{\frac{1}{2} + \gamma} \quad \forall i \in I \tag{25}$$

then choose $\epsilon^{k+1} \in (0, \eta \epsilon^k]$; else, set $\epsilon^{k+1} := \epsilon^k$.

Step 3. (Check inner loop stopping criteria)
If

$$\|x^{k+1} - x^k\|_2 \leq \sigma \quad \text{and} \quad \|\epsilon^k\|_2 \leq \sigma' \tag{26}$$

then proceed to Step 4; else, set $k := k+1$ and return to Step 1.

Step 4. (Update penalty parameters)
If $\|x^{j+1} - x^j\|_2 > \sigma$, then:

$$\begin{aligned} M_1^{j+1} &:= \beta M_1^j \\ M_2^{j+1} &:= \beta M_2^j \end{aligned} \tag{27}$$

where $\beta > 1$ is a penalty update factor (typically $\beta = 1.5$).

Step 5. (Check outer convergence)
If $j \geq 1$ and $\|x^{j+1} - x^j\|_2 \leq \sigma$, stop; else, set $j := j+1$ and return to Step 1.

D. Relationship with Original IRWA

The E-IRWA algorithm maintains the core advantages of the original IRWA method:

- Matrix-free implementation capability
- Use of projection operations for handling convex constraints
- Iterative weight updates based on constraint violation measures

The key enhancement lies in the introduction of the outer iteration loop that adaptively adjusts penalty parameters M_1 and M_2. This modification provides several advantages:

1) Different constraint types can be handled with appropriate penalty scales
2) The penalty parameters automatically adapt to problem characteristics
3) The nested iteration structure maintains numerical stability while ensuring constraint satisfaction

E. Implementation Considerations

The implementation of E-IRWA requires minimal modifications to existing IRWA code bases. The primary additions are:

- Storage for two penalty parameters instead of one
- An outer iteration loop for penalty parameter updates
- Modified weight calculations based on constraint type

The penalty update factor $\beta > 1$ (typically set to 1.5) provides a balance between aggressive constraint enforcement and numerical stability. The choice of initial penalty parameters M_1^0 and M_2^0 can affect the algorithm's early behavior but does not impact the final solution due to the adaptive nature of the updates.

XI. An Enhanced ADAL Method for Quadratic Programming with Mixed Constraints

A. Problem Formulation

We consider the quadratic programming problem with mixed ℓ_1 and one-sided constraints:

$$\min_{x \in \mathbb{R}^n} \quad \frac{1}{2} x^T H x + g^T x + \sum_{i \in I_1} |a_i^T x + b_i| + \sum_{i \in I_2} \max(a_i^T x + b_i, 0) \tag{28}$$

where $H \in \mathbb{R}^{n \times n}$ is symmetric positive semidefinite, $g \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and I_1, I_2 partition the constraint indices.

B. Enhanced ADAL Algorithm

We propose an enhanced ADAL algorithm that incorporates several numerical improvements:

1) Problem Reformulation: The problem is reformulated using auxiliary variables p:

$$\begin{aligned} \min_{x,p} \quad & \frac{1}{2} x^T H x + g^T x + \sum_{i \in I_1} |p_i| + \sum_{i \in I_2} \max(p_i, 0) \\ \text{s.t.} \quad & Ax + b = p \end{aligned} \tag{29}$$

2) Augmented Lagrangian: The augmented Lagrangian is:

$$\begin{aligned} L(x,p,u,\mu) = \frac{1}{2} x^T H x + g^T x + \sum_{i \in I_1} |p_i| + \sum_{i \in I_2} \max(p_i, 0) \\ + \frac{1}{2\mu} \|Ax + b - p + \mu u\|_2^2 - \frac{\mu}{2} \|u\|_2^2 \end{aligned} \tag{30}$$

The algorithm proceeds as follows:

Step 0. (Initialization)
Choose an initial point $x^0 \in \mathbb{R}^n$, auxiliary variable $p^0 \in \mathbb{R}^m$, dual variable $u^0 \in \mathbb{R}^m$, and penalty parameter $\mu >$

0. Let $\sigma \geq 0$ and $\sigma'' \geq 0$ be two scalars which serve as termination tolerances for the stepsize and constraint residual, respectively. Compute Cholesky factorization $LL^T = H + \mu A^T A$. Set outer iteration counter $j := 0$ and inner iteration counter $k := 0$.

Step 1. (Solve the x-subproblem)
Compute x^{k+1} by solving the linear system:

$$(H + \mu A^T A)x^{k+1} = -(g + A^T(\mu(b - p^k) + u^k))$$

using Cholesky factors:

(a) Solve $Ly = -(g + A^T(\mu(b - p^k) + u^k))$

(b) Solve $L^T x^{k+1} = y$ (31)

Step 2. (Solve the p-subproblem)
Given $q^k := Ax^{k+1} + b + u^k/\mu$, compute component-wise for p^{k+1}:
For indices $i \in I_1$:

$$p_i^{k+1} = q_i^k - \text{sign}(q_i^k) \max(|q_i^k| - \frac{1}{2\mu}, 0) \tag{32}$$

For indices $i \in I_2$:

$$p_i^{k+1} = q_i^k - \max(q_i^k - \frac{1}{2\mu}, 0) \tag{33}$$

For type-I constraints ($i \in I_1$), this update performs soft thresholding with the ℓ_1 penalty. For type-II constraints ($i \in I_2$), it applies the rectified linear unit (ReLU) like operation with smooth transition.

Step 3. (Update dual variables)
Set

$$u^{k+1} := u^k + \mu(Ax^{k+1} + b - p^{k+1}) \tag{34}$$

Step 4. (Check inner loop convergence)
Compute:

$$\begin{aligned} \text{residual} &:= \frac{\|Ax^{k+1} + b - p^{k+1}\|_2}{1 + \|b\|_2} \\ \text{step size} &:= \frac{\|x^{k+1} - x^k\|_2}{1 + \|x^k\|_2} \end{aligned} \tag{35}$$

If step size $\leq \sigma$ and residual $\leq \sigma''$, proceed to Step 5; else, set $k := k + 1$ and return to Step 1.

Step 5. (Penalty parameter update)
If residual $> 10\sigma''$, set:

$$\mu^{j+1} := \min(1.1\mu^j, 10^{10}) \tag{36}$$

and recompute Cholesky factorization $LL^T = H + \mu^{j+1} A^T A$.

Step 6. (Check outer loop convergence)
Compute relative change:

$$\text{rel\_change} := \frac{\|x^{j+1} - x^j\|_2}{1 + \|x^j\|_2} \tag{37}$$

If rel_change $\leq \sigma$, stop; else, set $j := j + 1$ and return to Step 1.

C. Key Algorithmic Improvements

1) Efficient Linear System Solution: We employ Cholesky factorization to efficiently solve the x-subproblem:

$$LL^T = H + \mu A^T A \tag{38}$$

This pre-computation significantly reduces the computational cost per iteration.

2) Adaptive Penalty Parameter: The penalty parameter μ is adjusted based on constraint violation:

$$\mu^{j+1} = \begin{cases} 1.1\mu^j & \text{if } \|Ax^{k+1} + b - p^{k+1}\|_2 > 10\sigma'' \\ \mu^j & \text{otherwise} \end{cases} \tag{39}$$

3) Robust Convergence Criteria: We employ relative error measures for more robust convergence detection:

$$\begin{aligned} \text{residual} &= \frac{\|Ax^{k+1} + b - p^{k+1}\|_2}{1 + \|b\|_2} \leq \sigma'' \\ \text{step size} &= \frac{\|x^{k+1} - x^k\|_2}{1 + \|x^k\|_2} \leq \sigma \end{aligned} \tag{40}$$

D. Numerical Considerations

The enhanced algorithm includes several numerical improvements:

- Threshold-based handling of small values in the p-subproblem
- Scaled constraint violations for better numerical stability
- Upper bound on penalty parameter growth to prevent overflow
- Relative error measures for convergence criteria

E. Implementation Efficiency

The implementation achieves efficiency through:

- Vectorized operations for constraint handling
- Pre-computed Cholesky factorization
- Adaptive parameter updates
- Early termination criteria

XII. Testing of the Algorithm

A. Review: real quadratic example

Quadratic Programming (QP) has established itself as a fundamental optimization framework with extensive real-world applications. Based on comprehensive literature reviews and recent developments, we present a systematic analysis of its applications across multiple domains.

TABLE VII: *Theoretical Computer Science Applications*

Application	Problem Characteristics	Key Constraints
Quadratic Assignment [25] [26]	Facility location mapping Flow optimization Discrete decision variables	Assignment constraints Flow conservation Binary variables
Maximum Cut [27] [28]	Graph partitioning Edge weight optimization NP-hard complexity	Cut-set constraints Binary decisions Balance requirements
Maximum Clique [29]	Graph theory application Vertex selection Discrete optimization	Adjacency constraints Clique size limits Binary selection

The theoretical computer science domain presents fundamental challenges in handling discrete optimization problems with combinatorial complexity. These problems typically require specialized solution methods that can effectively navigate large-scale binary decision spaces while maintaining computational efficiency.

TABLE VIII: *Computational Chemistry and Molecular Biology Applications*

Application	Problem Characteristics	Key Constraints
Zeolite Structure [30]	Molecular framework Energy minimization Atomic positioning	Bond length constraints Angle constraints Energy bounds
Molecular Dynamics [30]	Force field optimization Conformational analysis Potential energy surface	Physical constraints Energy conservation Molecular geometry

Applications in computational chemistry demand high precision and reliability in solution methods. The main challenge lies in balancing computational efficiency with solution accuracy, particularly when dealing with complex molecular structures and energy landscapes.

TABLE IX: *Financial Engineering Applications*

Application	Problem Characteristics	Key Constraints
Portfolio Optimization [31] [32] [33] [34] [35] [36]	Risk-return trade-off Large-scale optimization Multi-period planning	Budget constraints Risk limits Diversification rules
Risk Management [37] [38]	VaR optimization Scenario analysis Dynamic hedging	Risk metrics Regulatory limits Hedge ratios
Asset Allocation [35] [36]	Strategic investment Factor exposure Transaction costs	Capacity constraints Turnover limits Factor constraints

Financial applications require robust solution methods capable of handling large-scale problems with time-sensitive requirements. The complexity stems from market uncertainty, solution stability requirements, and the need for real-time decision making in dynamic market environments.

TABLE X: *Process Networks and Industrial Systems Applications*

Application	Problem Characteristics	Key Constraints
Oil Scheduling [39] [40] [41]	Time-indexed operations Resource allocation Multi-period planning	Flow balance Capacity limits Quality specifications
Multi -commodity Flow [42]	Network structure Resource sharing Flow optimization	Network flow conservation Capacity restrictions Quality requirements
Gas Networks [43] [44] [45]	Nonlinear dynamics Pressure-flow relations Network stability	Pressure bounds Flow conservation Safety requirements

Process networks present unique challenges due to their complex operational requirements and physical constraints. The optimization must account for both discrete operational decisions and continuous process variables while ensuring feasibility under varying conditions.

TABLE XI: *Supply Chain and Logistics Applications*

Application	Problem Characteristics	Key Constraints
Inventory Management [46]	Dynamic systems Demand uncertainty Multi-echelon structure	Storage capacity Service levels Cost optimization
Transportation Planning [42]	Network optimization Route selection Load balancing	Vehicle capacity Time windows Driver regulations
Warehouse Operations [46]	Resource allocation Order picking Storage assignment	Space constraints Labor availability Equipment utilization

Supply chain applications require robust optimization methods that can handle both strategic planning and operational decisions. The key challenge lies in managing uncertainty while maintaining operational efficiency across complex networks.

TABLE XII: *Computational Geometry Applications*

Application	Problem Characteristics	Key Constraints
Layout Design [47] [48] [49]	Spatial relationships Area optimization Placement decisions	Non-overlap constraints Aspect ratio limits Clearance requirements
Circle Packing [50] [51] [52] [53]	Container optimization Dense packing Pattern formation	Distance constraints Boundary conditions Stability requirements
Polygon Nesting [54] [55]	Material utilization Waste minimization Cutting patterns	Geometric feasibility Pattern constraints Material properties

Computational geometry applications focus on spatial optimization problems with complex geometric constraints. These problems often require specialized solution techniques to handle non-convex feasible regions and multiple local optima.

TABLE XIII: *Robotics and Control Applications*

Application	Problem Characteristics	Key Constraints
Trajectory Planning [56]	Dynamic systems Real-time control Path optimization	Motion constraints Collision avoidance Energy efficiency
Motion Control [56]	State feedback Stability requirements Performance optimization	Actuator limits Safety bounds Response time
Multi-robot Systems [56]	Coordination Task allocation Formation control	Inter-robot spacing Communication limits Resource sharing

Robotics applications demand fast, reliable solutions for real-time control and decision making. The primary challenges include handling dynamic constraints, ensuring safety, and maintaining computational efficiency under strict timing requirements.

TABLE XIV: *Energy Systems Applications*

Application	Problem Characteristics	Key Constraints
Unit Commitment [32] [34] [50] [57]	Binary commitment decisions Multi-period scheduling Cost minimization	Generation limits Ramping constraints Reserve requirements
Grid Optimization [57]	Network flow Voltage control Loss minimization	Power flow equations Stability constraints Security requirements
Demand Response [57]	Load shifting Price-based control Consumer behavior	User comfort Peak limitations Grid capacity

Energy system applications involve complex interactions between physical infrastructure, market dynamics, and operational constraints. The optimization must balance economic efficiency with system reliability and environmental considerations while meeting strict regulatory requirements.

TABLE XV: *Water Distribution Network Applications*

Application	Problem Characteristics	Key Constraints
Network Design [58] [59] [60]	Topology optimization Pipe sizing Cost minimization	Pressure requirements Flow conservation Water quality
Operation Control [61] [62] [63]	Pump scheduling Pressure management Quality control	Energy efficiency Service reliability Operational costs
Leakage Management [64] [65] [66]	Pressure optimization Zone creation Sensor placement	Detection accuracy Network resilience Maintenance costs

Water distribution applications require sophisticated optimization approaches that can handle both the physical dynamics of fluid networks and the practical constraints of infrastructure management. Key challenges include dealing with nonlinear hydraulic relationships and ensuring system reliability.

TABLE XVI: *Telecommunications Network Applications*

Application	Problem Characteristics	Key Constraints
Network Routing [67] [68]	Path optimization Traffic management QoS guarantees	Bandwidth constraints Delay requirements Capacity limits
Resource Allocation [67]	Frequency assignment Power control Channel scheduling	Interference limits Coverage requirements Service quality
Infrastructure Planning [68]	Network expansion Coverage optimization Capacity planning	Budget constraints Reliability targets Regulatory compliance

Telecommunications applications focus on optimizing network performance while managing complex resource allocation problems. The challenges include handling dynamic traffic patterns, ensuring quality of service, and meeting increasing bandwidth demands.

TABLE XVII: *Healthcare Resource Allocation Applications*

Application	Problem Characteristics	Key Constraints
Staff Scheduling [69]	Shift planning Skill matching Workload balancing	Coverage requirements Work regulations Staff preferences
Resource Utilization [69]	Equipment allocation Patient scheduling Facility planning	Capacity constraints Service times Emergency capacity
Supply Chain [69]	Inventory management Distribution planning Emergency response	Storage conditions Expiration dates Cost effectiveness

Healthcare applications require optimization methods that can balance multiple competing objectives while ensuring high service quality and resource efficiency. The optimization must account for uncertainty in demand and strict regulatory requirements.

TABLE XVIII: *Machine Learning and AI Applications*

Application	Problem Characteristics	Key Constraints
Model Training [70] [71]	Loss minimization Parameter optimization Feature selection	Generalization bounds Memory limitations Computational efficiency
Neural Networks [72]	Weight optimization Architecture search Hyperparameter tuning	Accuracy targets Training stability Resource constraints
Reinforcement Learning [70]	Policy optimization Value function estimation Exploration-exploitation	State-space constraints Action bounds Learning rate

Machine learning applications increasingly rely on quadratic programming for optimization tasks in training and model selection. The main challenges include handling large-scale datasets, ensuring model robustness, and managing computational resources efficiently.

TABLE XIX: *Smart Grid Operations*

Application	Problem Characteristics	Key Constraints
Microgrid Management [57]	Energy balance Renewable integration Load forecasting	Storage capacity Power quality Grid stability
Energy Storage [57]	Charging strategies Discharge planning Arbitrage optimization	Battery life Power limits Energy efficiency
Grid Resilience [57]	Fault management Contingency planning Risk mitigation	Recovery time System security Service continuity

Smart grid applications require sophisticated optimization methods to handle the increasing complexity of modern power systems. Key challenges include integrating renewable energy sources, managing distributed resources, and ensuring grid stability under uncertain conditions.

XIII. Numerical Experiments

A. Test Problems and Implementation

To comprehensively evaluate the performance of the proposed algorithms, we designed an extensive testing framework. All numerical experiments were conducted using Python 3.8 on a machine with Intel Core i7 processor and 16GB RAM.

1) Problem Generation: We generated quadratic programming (QP) problems with varying scales and characteristics as test instances. Each test instance contains the following components:

- Equality constraint matrix $A_1 \in \mathbb{R}^{m_{eq} \times n}$
- Inequality constraint matrix $A_2 \in \mathbb{R}^{m_{ineq} \times n}$
- Equality constraint vector $b_1 \in \mathbb{R}^{m_{eq}}$
- Inequality constraint vector $b_2 \in \mathbb{R}^{m_{ineq}}$
- Linear term vector in objective function $g \in \mathbb{R}^n$
- Quadratic term matrix in objective function $H \in \mathbb{R}^{n \times n}$

To ensure diversity and representativeness of the problems, we adopted the following generation strategy:

1) Matrix Generation: The elements of constraint matrices are generated from normal distributions with random means (range 1-10) and random variances (range 1-10).
2) Vector Generation: The elements of constraint vectors b are sampled from normal distributions with mean range [-100,100] and variance range [1,100].

3) Objective Function:
   - The linear term g is generated in the same way as constraint vectors
   - The quadratic term matrix H is constructed as $H = 0.1I + LL^T$, where L is a random normal distribution matrix, ensuring the positive definiteness of H

B. Benchmarking Framework

Our benchmarking framework includes the following key metrics:

- Computation Time: Recording the solving time for each solver on each instance
- Objective Function Value: Evaluating the quality of the final solution
- Constraint Violation:
  - Equality constraint violation: $\|A_1x - b_1\|$
  - Inequality constraint violation: $\|max(A_2x - b_2, 0)\|$
- Solution Success Rate: Statistical percentage of successfully solved instances for each solver

C. Experimental Settings

In the standard test configuration:

- Problem Dimension: $n = 100$ (number of decision variables)
- Number of Constraints:
  - $m_{eq} = 30$ (number of equality constraints)
  - $m_{ineq} = 30$ (number of inequality constraints)
- Number of Test Instances: 100 independently generated problem instances
- Solver Parameters:
  - IRWA Algorithm: $M_1 = M_2 = 10.0$, $max\_iter = 100$
  - ADAL Algorithm: $\mu = 100.0$, $\sigma = 10^{-3}$, $\sigma' = 10^{-4}$

D. Comparison with State-of-the-Art Solvers

To comprehensively evaluate the performance of the proposed algorithms, we selected 11 mainstream QP solvers as comparison benchmarks:

- clarabel: A new interior point method solver
- cvxopt: A Python software for convex optimization
- daqp: A dual active-set QP solver
- ecos: Embedded Conic Solver
- highs: High performance linear optimization solver
- osqp: Operator Splitting Quadratic Program solver
- piqp: Proximal Interior Point Quadratic Programming
- proxqp: Proximal Quasi-Newton Quadratic Programming
- qpalm: Quadratic Programming with Augmented Lagrangian Method
- quadprog: Quadratic Programming solver
- scs: Splitting Conic Solver

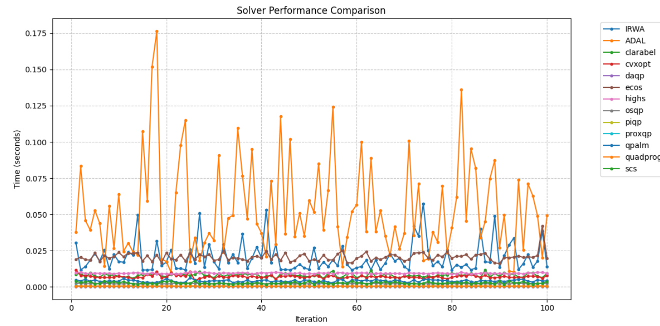

Fig. 2: *time analysis: the solving time comparison across 100 iterations for all tested solvers on standardized QP problems. The results demonstrate distinct performance patterns among different solver categories.*

E. Performance Metrics

For each solver, we collect and analyze:

- Mean and standard deviation of solving time
- Statistical distribution of objective function values
- Quantitative assessment of constraint satisfaction
- Algorithm convergence behavior and stability
- Performance on problems of different scales

The reproducibility of test results is guaranteed through fixed random seeds and standardized problem generation processes. This comprehensive testing framework evaluates not only the computational efficiency but also the solution quality and constraint satisfaction of the algorithms.

F. Computational Time Analysis

Fig. 2 presents the solving time comparison across 100 iterations for all tested solvers on standardized QP problems. The results demonstrate distinct performance patterns among different solver categories.

1) Performance Classification: Based on the computational time analysis, the solvers can be classified into three main categories:

- Fast Solvers ($< 0.01s$):
  - clarabel, daqp, piqp, and proxqp consistently demonstrate superior performance
  - Maintain stable solving times around 0.001-0.005 seconds
  - Show minimal variation across different problem instances
- Medium-Speed Solvers (0.01-0.03s):
  - cvxopt, highs, osqp, and scs exhibit moderate performance
  - Demonstrate consistent solving times between 0.01 and 0.03 seconds
  - Display good stability with minor variations
- Variable-Speed Solvers (0.03-0.18s):
  - Include IRWA, ADAL, and qpalm
  - Show significant variations in solving times
  - Display more sensitivity to problem characteristics

2) Algorithm-Specific Analysis: The proposed algorithms demonstrate distinct characteristics:

1) ADAL Algorithm:
   - Exhibits the highest variation in solving time
   - Maximum peaks reach approximately 0.175 seconds
   - Shows multiple performance spikes across iterations
   - Generally requires longer solving times compared to other solvers
2) IRWA Algorithm:
   - Demonstrates moderate performance variation
   - Solving times primarily range between 0.02-0.05 seconds
   - Shows better stability compared to ADAL
   - Maintains more consistent performance across iterations

3) Stability Analysis: The stability characteristics of different solver categories reveal important patterns:

- Commercial Solvers:
  - Demonstrate exceptional stability across iterations
  - Maintain consistent performance regardless of problem variations
  - Show minimal solving time fluctuations
- Proposed Algorithms:
  - Display higher sensitivity to problem characteristics
  - ADAL shows the most significant performance volatility
  - IRWA maintains better stability but still exhibits variation

4) Performance Implications: The computational time analysis reveals several key insights:

- The fastest commercial solvers consistently achieve sub-0.01 second solving times
- ADAL's variable performance suggests potential for algorithmic optimization
- IRWA demonstrates promising stability characteristics but requires efficiency improvements
- The mature implementation of commercial solvers results in more consistent performance

These results indicate that while both proposed algorithms successfully solve the QP problems, there exists significant potential for optimization to enhance their stability and reduce computational time to compete with leading commercial solvers.

XIV. Conclusion

This paper has presented a comprehensive examination of quadratic programming solvers and contributed enhanced implementations of two fundamental algorithms. Through systematic analysis and experimental validation, we have made several significant contributions to the field of optimization.

Our historical analysis has revealed three distinct generations in the evolution of QP solvers, from foundational implementations to current specialized solutions. This progression demonstrates the field's adaptation to emerging computational capabilities and application requirements. The systematic classification of solution methods has provided valuable insights into algorithmic development trends and specialization patterns, particularly highlighting the growing importance of hardware-specific optimization and real-time performance guarantees.

The algorithmic contributions of this work center on two key implementations. First, the enhanced IRWA implementation with adaptive penalty parameters has shown improved capabilities in handling heterogeneous constraints, demonstrating both stability and efficiency in solution computation. Second, our robust ADAL implementation for mixed constraints has exhibited competitive performance in managing complex optimization problems, particularly in terms of constraint satisfaction and solution quality.

Performance analysis through extensive numerical experiments has yielded several important findings. Commercial solvers, including clarabel, daqp, piqp, and proxqp, consistently achieve superior computational efficiency with solving times below 0.01 seconds. The proposed IRWA implementation demonstrates promising stability characteristics while maintaining moderate solving times between 0.02 and 0.05 seconds. The ADAL implementation, while showing higher variation in solving times, proves particularly robust in constraint satisfaction.

The comprehensive review of application domains reveals increasing complexity in problem requirements across various sectors. This is particularly evident in emerging fields such as machine learning and smart grid operations, where optimization problems present new challenges and opportunities. The analysis clearly establishes the need for specialized implementations targeting specific application requirements.

Looking ahead, several promising research directions emerge from this work. Hardware acceleration and GPU utilization present significant opportunities for performance improvement. The integration of machine learning techniques for adaptive solver selection and parameter tuning shows promise for enhancing solver robustness. Additionally, the development of more sophisticated handling of ill-conditioned problems and numerical stability remains an important area for future research.

Our findings suggest that future developments should focus on further optimization of solver implementations for specific hardware architectures, development of more sophisticated warm-starting techniques for real-time applications, and adaptation of algorithms for distributed and edge computing environments. The comprehensive analysis provided in this paper offers practical guidance for practitioners while establishing a foundation for future research in quadratic programming optimization.

Announcement about the generative AI

In the course of this research, we utilized Chatgpt to assist in the literature review process. This tool was instrumental in summarizing information and clarifying relevant professional knowledge, thereby enriching the content of the review. It is important to note that the use of generative AI was limited to this specific aspect of the research. The other core sections of the paper were developed without the aid of generative AI, ensuring that the analysis, interpretations, and conclusions presented reflect our own scholarly efforts and insights.

REFERENCES

[1] Burke, James V., et al. "Iterative reweighted linear least squares for exact penalty subproblems on product sets." SIAM Journal on Optimization 25.1 (2015): 261-294.

[2] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," SIAM Journal on Optimization, vol. 12, no. 4, pp. 979-1006, 2002.

[3] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," Mathematical Programming, March 2004.

[4] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Knitro: An Integrated Package for Nonlinear Optimization," Large-Scale Nonlinear Optimization, February 2006.

[5] L. Vandenberghe, "The CVXOPT linear and quadratic cone program solvers," Technical Report, March 2010.

[6] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP Solver for Embedded Systems," in European Control Conference (ECC), 2013.

[7] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," Mathematical Programming Computation, 2014.

[8] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding," Journal of Optimization Theory and Applications, July 2016.

[9] Q. Huangfu and J. A. J. Hall, "Parallelizing the dual revised simplex method," Mathematical Programming Computation, vol. 9, December 2017.

[10] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," Mathematical Programming Computation, January 2018.

[11] A. G. Pandala, Y. Ding, and H. Park, "qpSWIFT: A Real-Time Sparse Quadratic Program Solver for Robotic Applications," 2019.

[12] B. A. Turlach, A. Weingessel, and C. Moler, "Functions to Solve Quadratic Programming Problems," November 2019.

[13] N. Sakre, A. Habe, A. R. Pettitt, and T. Okamoto, "Massive core/star formation triggered by cloud-cloud collision: Effect of magnetic field," The Astrophysical Journal, 2020.

[14] Y. Jo, J. Visser, C. Reed, and E. Hovy, "Extracting Implicitly Asserted Propositions in Argumentation," 2020.

[15] D. Arnström, A. Bemporad, and D. Axehill, "A Dual Active-Set Solver for Embedded Quadratic Programming Using Recursive LDLT Updates," arXiv preprint arXiv:2103.16236v2, March 2021.

[16] R. Luce, "Quadratic optimization," Gurobi Optimization Technical Report, October 2022.

[17] R. Schwan, Y. Jiang, D. Kuhn, and C. N. Jones, "PIQP: A Proximal Interior-Point Quadratic Programming Solver," EPFL Technical Report, September 2023.

[18] A. Bambade, F. Schramm, S. El Kazdadi, S. Caron, A. Taylor, and J. Carpentier, "PROXQP: an Efficient and Versatile Quadratic Programming Solver for Real-Time Robotics Applications and Beyond," 2023.

[19] A. L. Bishop, J. Z. Zhang, S. Gurumurthy, K. Tracy, and Z. Manchester, "ReLU-QP: A GPU-Accelerated Quadratic Programming Solver for Model-Predictive Control," November 2023.

[20] P. J. Goulart and Y. Chen, "Clarabel: An interior-point solver for conic programs with quadratic objectives," Journal of Optimization Theory and Applications, May 2024.

[21] IBM, "Solving problems with a quadratic objective (QP)," IBM Technical Report, 2024.

[22] K. Tracy and Z. Manchester, "On the Differentiability of the Primal-Dual Interior-Point Method," Carnegie Mellon University, June 2024.

[23] MOSEK ApS, "MOSEK Modeling Cookbook," Release 3.3.0, September 2024.

[24] W. Gu et al., "Identifying Performance Issues in Cloud Service Systems Based on Relational-Temporal Features," November 2024.

[25] K. M. Anstreicher, "Recent advances in the solution of quadratic assignment problems," Mathematical Programming, November 2003.

[26] E. M. Loiola et al., "A survey for the quadratic assignment problem," European Journal of Operational Research, November 2007.

[27] N. Krislock et al., "BiqCrunch: A semidefinite branch-and-bound method for solving binary quadratic problem," ACM Transactions on Mathematical Software, January 2017.

[28] F. Rendl et al., "Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations," Mathematical Programming, November 2008.

[29] I. M. Bomze et al., "The maximum clique problem," Handbook of Combinatorial Optimization, November 1999.

[30] C. E. Gounaris et al., "Estimation of diffusion anisotropy in microporous crystalline materials and optimization of crystal orientation in membranes," The Journal of Chemical Physics, September 2013.

[31] Z. Deng et al., "A branch-and-cut approach to portfolio selection with marginal risk control in a linear conic programming framework," Journal of Systems Science and Systems Engineering, December 2013.

[32] A. Frangioni and C. Gentile, "Perspective Cuts for a Class of Convex 0–1 Mixed Integer Programs," Mathematical Programming, April 2006.

[33] A. Frangioni and C. Gentile, "SDP Diagonalizations and Perspective Cuts for a Class of Nonseparable MIQP," Operations Research Letters, March 2007.

[34] A. Frangioni and C. Gentile, "A Computational Comparison of Reformulations of the Perspective Relaxation: SOCP vs. Cutting Planes," Operations Research Letters, May 2009.

[35] J. Kallrath, "Exact computation of global minima of a nonconvex portfolio optimization problem," Frontiers in Global Optimization, 2003.

[36] X. Lin et al., "Global solution approach for a nonconvex MINLP problem in product portfolio optimization," Journal of Global Optimization, November 2005.

[37] C. D. Maranas et al., "Solving long-term financial planning problems via global optimization," Journal of Economic Dynamics and Control, August 1997.

[38] P. Parpas and B. Rustem, "Global optimization of the scenario generation and portfolio selection problems," Computational Science and Its Applications, May 2006.

[39] J. Li et al., "Improving the robustness and efficiency of crude scheduling algorithms," AIChE Journal, October 2007.

[40] J. Li et al., "Continuous-time modeling and global optimization approach for scheduling of crude oil operations," AIChE Journal, January 2012.

[41] J. Li et al., "Scheduling of crude oil operations under demand uncertainty: A robust optimization framework coupled with global optimization," AIChE Journal, August 2012.

[42] B. Tadayon and J. C. Smith, "Algorithms for an integer multicommodity network flow problem with node reliability considerations," Journal of Optimization Theory and Applications, May 2013.

[43] M. M. F. Hasan et al., "Preliminary synthesis of fuel gas networks to conserve energy and preserve the environment," Industrial & Engineering Chemistry Research, June 2011.

[44] X. Li et al., "Stochastic pooling problem for natural gas production network design and operation under uncertainty," AIChE Journal, August 2011.

[45] X. Li et al., "Decomposition strategy for the stochastic pooling problem," Journal of Global Optimization, April 2012.

[46] A. Nyberg et al., "The optimal design of a three-echelon supply chain with inventories under uncertainty," 2012.

[47] M. F. Anjos and F. Liers, "Global approaches for facility layout and VLSI floorplanning," Handbook on Semidefinite, Conic and Polynomial Optimization, 2012.

[48] I. Castillo et al., "Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods," Computers & Chemical Engineering, January 2005.

[49] M. C. Dorneich and N. V. Sahinidis, "Global optimization algorithms for chip layout and compaction," Engineering Optimization, 1995.

[50] A. Frangioni et al., "Approximated Perspective Relaxations: a Project & Lift Approach," Computational Optimization and Applications, April 2016.

[51] A. Frangioni et al., "Projected Perspective Reformulations with Applications in Design Problems," Operations Research, September 2011.

[52] M. Hifi and R. M'Hallah, "A literature review on circle and sphere packing problems: Models and methodologies," Advances in Operations Research, 2009.

[53] P. G. Szabó et al., “Global optimization in geometry – circle packing into the square,” Essays and Surveys in Global Optimization, 2005.
[54] J. Kallrath, “Cutting circles and polygons from area-minimizing rectangles,” Journal of Global Optimization, March 2009.
[55] S. Rebennack et al., “Column enumeration based decomposition techniques for a class of non-convex MINLP problems,” Journal of Global Optimization, 2009.
[56] I. Gentilini et al., “The travelling salesman problem with neighbourhoods: MINLP solution,” Optimization Methods and Software, April 2013.
[57] M. Tahanan et al., “Large-scale Unit Commitment under uncertainty,” 4OR, June 2015.
[58] E. Ahmetović and I. E. Grossmann, “Global superstructure optimization for the design of integrated process water networks,” AIChE Journal, February 2011.
[59] M. Bagajewicz, “A review of recent design procedures for water networks in refineries and process plants,” Computers & Chemical Engineering, September 2000.
[60] C. Bragalli et al., “On the optimal design of water distribution networks: A practical MINLP approach,” Optimization and Engineering, June 2012.
[61] P. M. Castro and J. P. Teles, “Comparison of global optimization algorithms for the design of water-using networks,” Computers & Chemical Engineering, April 2013.
[62] B. Geissler et al., “A new algorithm for MINLP applied to gas transport energy cost minimization,” Facets of Combinatorial Optimization, 2013.
[63] A. M. Gleixner et al., “Towards globally optimal operation of water supply networks,” Numerical Algebra, Control and Optimization, December 2012.
[64] J. Jeżowski, “Review of water network design methods with literature annotations,” Industrial & Engineering Chemistry Research, 2010.
[65] C. S. Khor et al., “Fixed-flowrate total water network synthesis under uncertainty with risk management,” Journal of Cleaner Production, 2014.
[66] J. M. Ponce-Ortega et al., “Global optimization for the synthesis of property-based recycle and reuse networks including environmental constraints,” Computers & Chemical Engineering, March 2010.
[67] A. Frangioni et al., “Delay-Constrained Shortest Paths: Approximation Algorithms and Second-Order Cone Models,” Journal of Optimization Theory and Applications, March 2015.
[68] A. Frangioni et al., “Delay-constrained routing problems: Accurate scheduling models and admission control,” Computers & Operations Research, 2017.
[69] J. Castro et al., “Perspective Reformulations of the CTA Problem with L2 Distances,” Operations Research, August 2014.
[70] G. Di Pillo et al., “A class of structured quasi-newton algorithms for optimal control problems,” IFAC Proceedings Volumes, June 1983.
[71] K. Schittkowski, “Numerical solution of a time-optimal parabolic boundary-value control problem,” Journal of Optimization Theory and Applications, February 1979.
[72] S. Stojanovic, “Optimal damping control and nonlinear elliptic systems,” SIAM Journal on Control and Optimization, May 1991.