Computational Optimization and Applications, 18, 295–309, 2001

Integrating SQP and Branch-and-Bound for Mixed Integer Nonlinear Programming

SVEN LEYFFER sleyffer@mcs.dundee.ac.uk
Department of Mathematics, University of Dundee, Dundee, U.K.

Received November 9, 1998; Accepted August 31, 1999

Abstract. This paper considers the solution of Mixed Integer Nonlinear Programming (MINLP) problems. Classical methods for the solution of MINLP problems decompose the problem by separating the nonlinear part from the integer part. This approach is largely due to the existence of packaged software for solving Nonlinear Programming (NLP) and Mixed Integer Linear Programming problems.

In contrast, an integrated approach to solving MINLP problems is considered here. This new algorithm is based on branch-and-bound, but does not require the NLP problem at each node to be solved to optimality. Instead, branching is allowed after each iteration of the NLP solver. In this way, the nonlinear part of the MINLP problem is solved whilst searching the tree. The nonlinear solver that is considered in this paper is a Sequential Quadratic Programming solver.

A numerical comparison of the new method with nonlinear branch-and-bound is presented and a factor of up to 3 improvement over branch-and-bound is observed.

Keywords: mixed integer nonlinear programming, branch-and-bound, sequential quadratic programming

1. Introduction

This paper considers the solution of Mixed Integer Nonlinear Programming (MINLP) problems. Problems of this type arise when some of the variables are required to take integer values. Applications of MINLP problems include the design of batch plants (e.g. [17] and [21]), the synthesis of processes (e.g. [7]), the design of distillation sequences (e.g. [27]), the optimal positioning of products in a multi-attribute space (e.g. [7]), the minimization of waste in paper cutting [29] and the optimization of core reload patterns for nuclear reactors [25]. For a comprehensive survey of MINLP applications see Grossmann and Kravanja [16].

MINLP problems can be modelled in the following form

$$(P) \quad \begin{cases} \underset{x,y}{\text{minimize}} & f(x, y) \\ \text{subject to} & g(x, y) \leq 0 \\ & x \in X, y \in Y \text{ integer}, \end{cases}$$

where y are the integer variables modelling for instance decisions ($y \in \{0, 1\}$), numbers of equipment ($y \in \{0, 1, \ldots, N\}$) or discrete sizes and x are the continuous variables.

Classical methods for the solution of (P) "decompose" the problem by separating the nonlinear part from the integer part. This approach is made possible by the existence of packaged software for solving Nonlinear Programming (NLP) and Mixed Integer Linear Programming (MILP) problems. The decomposition is even more apparent in Outer Approximation ([7] and [10]) and Benders Decomposition ([15] and [13]) where an alternating sequence of MILP master problems and NLP subproblems obtained by fixing the integer variables is solved (see also the recent monograph of Floudas [14]). The recent branch-and-cut method for 0-1 convex MINLP of Stubbs and Mehrotra [26] also separates the nonlinear (lower bounding and cut generation) and integer part of the problem. Branch-and-bound [5] can also be viewed as a "decomposition" in which the tree-search (integer part) is largely separated from solving the NLP problems at each node (nonlinear part).

In contrast to these approaches, the present paper considers an integrated approach to MINLP problems. The algorithm developed here is based on branch-and-bound, but instead of solving an NLP problem at each node of the tree, the tree-search and the iterative solution of the NLP are interlaced. Thus the nonlinear part of (P) is solved whilst searching the tree. The nonlinear solver that is considered in this paper is a Sequential Quadratic Programming (SQP) solver. SQP solves an NLP problem via a sequence of quadratic programming (QP) problem approximations.

The basic idea underlying the new approach is to branch early—possibly after a single QP iteration of the SQP solver. This idea was first presented by Borchers and Mitchell [1]. The present paper improves on their idea in a number of ways:

1. In order to derive lower bounds needed in branch-and-bound Borchers and Mitchell propose to evaluate the Lagrangian dual. This is effectively an unconstrained nonlinear optimization problem. In Section 3 it is shown that there is no need to compute a lower bound if the linearizations in the SQP solver are interpreted as cutting planes.
2. In [1] *two* QP problems are solved before branching. This restriction is due to the fact that a packaged SQP solver is used. By using our own solver we are able to remove this unnecessary restriction, widening the scope for early branching.
3. A new heuristic for deciding when to branch early is introduced which does not rely on a *fixed* parameter, but takes the second order rate of convergence of the SQP solver into account when deciding whether to branch or continue with the SQP iteration.

The ideas presented in this paper have a similar motivation as a recent algorithm by Quesada and Grossmann [24]. Their algorithm is related to outer approximation but avoids the resolution of related MILP master problems by interrupting the MILP branch-and-bound solver each time an integer node is encountered. At this node an NLP problem is solved (obtained by fixing all integer variables) and new outer approximations are added to all problems on the MILP branch-and-bound tree. Thus the MILP is updated and the tree-search resumes. The difference to the approach considered here is that Quesada and Grossmann still solve NLP problems at some nodes, whereas the new solver presented here usually solves one quadratic programming (QP) problem at each node.

The paper is organized as follows. Section 2 briefly reviews the SQP method and branch-and-bound. In Section 3 the new algorithm is developed for the special case of convex

MINLP problems. Section 3.4 introduces a heuristic for handling nonconvex MINLP problems. Finally, Section 4 presents a comparison of the new method with nonlinear branch-and-bound. These results are very encouraging, often improving on branch-and-bound by a factor of up to 3.

2. Background

2.1. *Sequential quadratic programming*

A popular iterative method for solving NLP problems is Sequential Quadratic Programming (SQP). The basic form of SQP methods date back to Wilson [30] and were popularized by Han [19] and Powell [23], see Fletcher [9, Chapter 12.4] and Conn, Gould and Toint [4] for a selection of other references. In its basic form, SQP solves an NLP problem via a sequence of quadratic programming (QP) approximations obtained by replacing the nonlinear constraints by a linear first order Taylor series approximation and the nonlinear objective by a second order Taylor series approximation augmented by second order information from the constraints. It can be shown under certain conditions that the SQP method converges quadratically near a solution.

It is well known that the SQP method may fail to converge if it is started far from a local solution. In order to induce convergence, many popular methods use a *penalty function* which is a linear combination of the objective function f and some measure of the constraint violation. A related idea is an *augmented Lagrangian function* in which a weighted penalty term is added to a Lagrangian function. A step in an SQP method is then accepted when it produces a sufficient decrease in the penalty function.

Two frameworks exist which enforce sufficient decrease, namely line-search in the direction of the QP solution or a trust-region that limits the QP step that is computed (see e.g. [9, Chapter 12.4] and references therein). In our implementation global convergence is promoted through the use of a trust-region and the new concept of a "filter" [11] which accepts a trial point whenever the objective or the constraint violation is improved compared to all previous iterates. The size of the trust-region is reduced if the step is rejected and increased if it is accepted.

2.2. *Branch-and-bound*

Branch-and-bound dates back to Land and Doig [22]. The first reference to nonlinear branch-and-bound can be found in Dakin [5]. It is most conveniently explained in terms of a tree-search.

Initially, all integer restrictions are relaxed and the resulting NLP relaxation is solved. If all integer variables take an integer value at the solution then this solution also solves the MINLP. Usually, some integer variables take a non-integer value. The algorithm then selects one of those integer variables which take a non-integer value, say y_i, with value $\hat{y}_i$, and branches on it. Branching generates two new NLP problems by adding simple bounds $y_i \leq [\hat{y}_i]$ and $y_i \geq [\hat{y}_i] + 1$ respectively to the NLP relaxation (where $[a]$ is the largest integer not greater than a).

One of the two new NLP problems is selected and solved next. If the integer variables take non-integer values then branching is repeated, thus generating a branch-and-bound tree whose nodes correspond to NLP problems and where an edge indicates the addition of a branching bound. If one of the following fathoming rules is satisfied, then no branching is required, the corresponding node has been fully explored (fathomed) and can be abandoned. The fathoming rules are

FR1 An infeasible node is detected. In this case the whole subtree starting at this node is infeasible and the node has been fathomed.

FR2 An integer feasible node is detected. This provides an upper bound on the optimum of the MINLP; no branching is possible and the node has been fathomed.

FR3 A lower bound on the NLP solution of a node is greater or equal than the current upper bound. In this case the node is fathomed, since this NLP solution provides a lower bound for all problems in the corresponding sub-tree.

Once a node has been fathomed the algorithm backtracks to another node which has not been fathomed until all nodes are fathomed.

Many heuristics exist for selecting a branching variable and for choosing the next problem to be solved after a node has been fathomed (see surveys by Gupta and Ravindran [18] and Volkovich et al. [28]).

Branch-and-bound can be inefficient in practice since it requires the solution of one NLP problem per node which is usually solved iteratively through a sequence of quadratic programming problems. Moreover, a large number of NLP problems are solved which have often no physical meaning if the integer variables do not take integer values.

3. Integrating SQP and branch-and-bound

An alternative to nonlinear branch-and-bound for *convex* MINLP problems is due to Borchers and Mitchell [1]. They observe that it is not necessary to solve the NLP at each node to optimality before branching and propose an *early branching rule*, which branches on an integer variable before the NLP has converged. Borchers and Mitchell implement this approach and report encouraging numerical results compared to Outer Approximation [2]. In their implementation the SQP solver is interrupted after two QP solves[1] and branching is done on an integer variable which is more than a given tolerance away from integrality.

The drawback of the *early branching rule* is that since the NLP problems are *not* solved to optimality, there is no guarantee that the current value of $f(x, y)$ is a lower bound. As a consequence, bounding (fathoming rule **FR3**) is no longer applicable. Borchers and Mitchell seek to overcome this difficulty by evaluating the Lagrangian dual for a given set of multiplier estimates, λ. The evaluation of the Lagrangian dual amounts to solving

$$\begin{cases} \underset{x,y}{\text{minimize}} & f(x, y) + \lambda^T g(x, y) \\ \text{subject to} & x \in X, y \in \hat{Y} \end{cases}$$

where the $\hat{Y} \subset Y$ now also includes bounds that have been added during the branching.

Note that this evaluation requires the solution of a bound constrained nonlinear optimization problem. In order to diminish the costs of these additional calculations, Borchers and Mitchell only evaluate the dual every sixth QP solve.

Clearly, it would be desirable to avoid the solution of this problem and at the same time obtain a lower bounding property at each QP solve. In the remainder of this section it is shown how this can be achieved by integrating the iterative NLP solver with the tree search.

Throughout this section it is assumed that both f and g are smooth, *convex* functions. The nonconvex case is discussed in Section 3.4 where a simple heuristic is proposed. The ideas are first presented in the context of a basic SQP method *without* globalization strategy. Next this basic algorithm is modified to allow for the use of a line-search or a trust-region. It is then shown how the early branching rule can take account of the quadratic rate of convergence of the SQP method. Finally, a simple heuristic for nonconvex MINLP problems is suggested.

3.1. The basic SQP algorithm

This section shows how the solution of the Lagrangian dual can be avoided by interpreting the constraints of the QP approximation as supporting hyperplanes. If f and g are convex, then the linear approximations of the QP are outer approximations and this property is exploited to replace the lower bounding.

Consider an NLP problem at a given node of the branch-and-bound tree,

$$(\hat{P}) \quad \begin{cases} \underset{x,y}{\text{minimize}} & f(x, y) \\ \text{subject to} & g(x, y) \leq 0 \\ & x \in X, y \in \hat{Y}, \end{cases}$$

where the integer restrictions have been relaxed and $\hat{Y} \subset Y$ contains bounds that have been added by the branching. Let $\hat{f}$ denote the solution of $(\hat{P})$.

Applying the SQP method to $(\hat{P})$ results in solving a sequence of QP problems of the form

$$(QP^k) \quad \begin{cases} \underset{d}{\text{minimize}} & f^{(k)} + \nabla f^{(k)^T} d + \frac{1}{2} d^T W^{(k)} d \\ \text{subject to} & g^{(k)} + \nabla g^{(k)^T} d \leq 0 \\ & x^k + d_x \in X, y^k + d_y \in \hat{Y}. \end{cases}$$

where $f^{(k)} = f(x^{(k)}, y^{(k)})$ etc. and

$$W^{(k)} \simeq \nabla^2 \mathcal{L}^{(k)} = \nabla^2 f^{(k)} + \sum \lambda_i \nabla^2 g_i^{(k)}$$

approximates the Hessian of the Lagrangian, where λ_i is the Lagrange multiplier of $g_i(x, y) \leq 0$.

Unfortunately, the solution of (QP^k) does not underestimate $\hat{f}$, even if all functions are assumed to be convex. This implies that fathoming rule **FR3** cannot be used if early

branching is employed. However, it turns out that it is not necessary to compute an underestimator explicitly. Instead, a mechanism is needed of terminating the sequence of (QP^k) problems once such an underestimator would become greater than the current upper bound, U, of the branch-and-bound process. This can be achieved by adding the *objective cut*

$$f^{(k)} + \nabla f^{(k)^T} d \leq U - \epsilon$$

to (QP^k), where $\epsilon > 0$ is the optimality tolerance of branch-and-bound. Denote the new QP with this objective cut by (QP_c^k).

$$\left(QP_c^k\right) \quad \begin{cases} \underset{d}{\text{minimize}} & f^{(k)} + \nabla f^{(k)^T} d + \dfrac{1}{2} d^T W^{(k)} d \\ \text{subject to} & g^{(k)} + \nabla g^{(k)^T} d \leq 0 \\ & f^{(k)} + \nabla f^{(k)^T} d \leq U - \epsilon \\ & x^k + d_x \in X, y^k + d_y \in \hat{Y}. \end{cases}$$

Note that (QP_c^k) is the QP that is generated by the SQP method when applied to the following NLP problem

$$\begin{cases} \underset{x,y}{\text{minimize}} & f(x, y) \\ \text{subject to} & g(x, y) \leq 0 \\ & f(x, y) \leq U - \epsilon \\ & x \in X, y \in \hat{Y}, \end{cases}$$

where a nonlinear objective cut has been added to $(\hat{P})$. This NLP problem is infeasible, if fathoming rule **FR3** holds at the solution to $(\hat{P})$.

It is now possible to replace fathoming rule **FR3** applied to $(\hat{P})$ by a test for feasibility of (QP_c^k) (replacing the bounding in branch-and-bound). This results in the following lemma.

Lemma 1. *Let f and g be smooth, convex functions. A sufficient condition for fathoming rule* **FR3** *applied to $(\hat{P})$ to be satisfied is that any QP problem (QP_c^k) generated by the SQP method in solving $(\hat{P})$ is infeasible.*

Proof: If (QP_c^k) is infeasible, then it follows that there exists no step d such that

$$f^{(k)} + \nabla f^{(k)^T} d \leq U - \epsilon \tag{1}$$

$$g^{(k)} + \nabla g^{(k)^T} d \leq 0. \tag{2}$$

Since (2) is an outer approximation of the nonlinear constraints of $(\hat{P})$ and (1) underestimates f, it follows that there exists no point $(\hat{x}, \hat{y}) \in \boldsymbol{X} \times \hat{\boldsymbol{Y}}$ such that

$$f(\hat{x}, \hat{y}) \leq U - \epsilon.$$

Thus any lower bound on f at the present node has to be larger than $U - \epsilon$. Thus to within the tolerance ϵ, $f \geq U$ and fathoming rule **FR3** holds. □

In practice, an SQP method would use a primal active set QP solver which establishes feasibility first and then optimizes. Thus the test of Lemma 1 comes at no extra cost. The basic integrated SQP and branch-and-bound algorithm can now be stated.

Algorithm 1: Basic SQP and branch-and-bound
Initialization: Place the continuous relaxation of (P) on the tree and set $U = \infty$.

while (there are pending nodes in the tree) **do**
Select an unexplored node.
repeat (SQP iteration)
1. Solve (QP_c^k) for a step $d^{(k)}$ of the SQP method.
2. **if** ((QP_c^k) infeasible) **then** fathome node, **exit**
3. Set $(x^{(k+1)}, y^{(k+1)}) = (x^{(k)}, y^{(k)}) + (d_x^{(k)}, d_y^{(k)})$.
4. **if** ($(x^{(k+1)}, y^{(k+1)})$ NLP optimal) **then**
   if ($x^{(k+1)}$ integral) **then**
   Update current best point by setting
   $(x^*, y^*) = (x^{(k+1)}, y^{(k+1)})$, $f^* = f^{(k+1)}$ and $U = f^*$.
   else
   Choose a non-integral $y_i^{(k+1)}$ and branch.
   endif
   exit
   endif
5. Compute the integrality gap $\theta = \max_i |y_i^{(k+1)} - \text{anint}(y_i^{(k+1)})|$.
6. **if** ($\theta > \tau$) **then**
   Choose a non-integral $y_i^{(k+1)}$ and branch, **exit**
   endif
end while

Here anint(y) is the nearest integer to y. Step 2 implements the fathoming rule of Lemma 1 and Step 6 is the early branching rule. In [1] a value of $\tau = 0.1$ is suggested for the early branching rule and this value has also been chosen here.

A convergence proof for this algorithm is given in the next section where it is shown how the algorithm has to be modified if a line-search or a trust-region is used to enforce global convergence for SQP. The key idea in the convergence proof is that (as in nonlinear branch-and-bound), the union of the child problems that are generated by branching is equivalent to the parent problem. The fact, that only a single QP has been solved in the parent problem does not alter this equivalence. Finally, the new fathoming rule implies **FR3** and hence, any node that has been fathomed by Algorithm 1 can also be considered as having been fathomed by nonlinear branch-and-bound. Thus convergence follows from the convergence of branch-and-bound.

Algorithm 1 has two important advantages over the work in [1]. Firstly, the lower bounding can be implemented at no additional cost (compared to the need to solve the Lagrangian dual in [1]). Secondly, the lower bounding is available at *all* nodes of the branch-and-bound tree (rather than at every sixth node). Note that if no further branching is possible at a node, then the algorithm resorts to normal SQP at that node.

3.2. The globalized SQP algorithm

It is well known that the basic SQP method may fail to converge if started far from a solution. In order to obtain a globally convergent SQP method, descent in some merit function has to be enforced. There are two concepts which enforce descent: a line-search and a trust-region. This section describes how the basic algorithm of the previous section has to be modified to accommodate these global features.

The use of a line-search requires only to replace $d^{(k)}$ by $\alpha^{(k)}d^{(k)}$, where $\alpha^{(k)}$ is the step-length chosen in the line-search. Similarly a Levenberg-Marquardt approach would require minimal change to the algorithm.

An alternative to using a line-search is to use a trust region. Trust-region SQP methods usually possess stronger convergence properties and this leads us to explore this approach. In this type of approach the step that is computed in (QP_c^k) is restricted to lie within a trust-region. Thus the QP that is being solved at each iteration becomes

$$\left(QP^k_{c,\infty}\right) \quad \begin{cases} \underset{d}{\text{minimize}} & f^{(k)} + \nabla f^{(k)^T} d + \frac{1}{2} d^T W^{(k)} d \\ \text{subject to} & g^{(k)} + \nabla g^{(k)^T} d \leq 0 \\ & f^{(k)} + \nabla f^{(k)^T} d \leq U - \epsilon \\ & \|d\|_\infty \leq \rho^k \\ & x^{(k)} + d_x \in X, y^k + d_y \in \hat{Y} \end{cases}$$

where ρ^k is the trust-region radius which is adjusted according to how well the quadratic model of $(QP^k_{c,\infty})$ agrees with the true function (see [9, Chapter 14.5] for details).

The drawback of using a trust-region in the present context is that the trust-region may cause $(QP^k_{c,\infty})$ to be infeasible even though the problem without a trust-region (QP^k_c) has a non-empty feasible region. Thus Lemma 1 is no longer valid. However, if $(QP^k_{c,\infty})$ is infeasible and the trust-region is inactive, then the argument of Lemma 1 still holds and the node can be fathomed.

Otherwise, the *feasibility restoration phase* has to be entered and this part of the SQP method is briefly described next. The aim of the restoration phase is to get closer to the feasible region by minimizing the violation of the constraints in some norm, subject to the linear constraints $x \in X,\ y \in \hat{Y}$.

$$(F) \quad \begin{cases} \underset{x,y}{\text{minimize}} & \|h^+(x,y)\| \\ \text{subject to} & x \in X, y \in \hat{Y}, \end{cases}$$

where

$$h(x, y) = \begin{pmatrix} f(x, y) - (U - \epsilon) \\ g(x, y) \end{pmatrix}$$

and the operation $a^+ = \max(0, a)$ is taken componentwise.

In this phase, an SQP algorithm is applied to minimize (F). Methods for solving this problem include Yuan [31] for the l_∞ norm, El-Hallabi and Tapia [8] for the l_2 norm and Dennis, El-Alem and Williamson [6] for the l_1 norm. These methods converge under mild assumptions similar to those needed or the convergence of SQP methods. The details of the restoration phase are beyond the scope of this paper and we concentrate instead on the interaction between this phase and branch-and-bound. In the remainder it will be assumed that this phase converges.

In order to obtain an efficient MINLP solver it is important to also integrate the restoration phase with the tree-search. In the restoration phase, a sequence of phase I type QP problems is solved. After each such QP four possible outcomes arise.

1. $(QP^k_{c,\infty})$ is feasible. In this case the solver exits the restoration phase and resumes SQP and early branching.
2. $(QP^k_{c,\infty})$ is not feasible and a step $d^{(k)}$ is computed for which the trust-region is inactive $(\|d^{(k)}\| < \rho^k)$. This indicates that the current node can be fathomed by Lemma 1.
3. The feasibility restoration converges to a minimum of the constraint violation (x^*, y^*) with $\|h^+(x^*, y^*)\| > 0$. This is taken as an indication that $(\hat{P})$ has no feasible point with an objective that is lower than the current upper bound and the corresponding node can be fathomed. Note that convergence of the feasibility SQP ensures that the trust-region will be inactive at (x^*, y^*).
4. The feasibility restoration SQP continues if none of the above occur.

Thus the feasibility restoration can be terminated before convergence, if 1 or 2 hold above. In the first case early branching resumes and in the second case, the node can be fathomed.

The modifications required to make Algorithm 1 work for trust-region SQP methods are given below.

Algorithm 2: Trust-region SQP and branch-and-bound

Only steps 1 and 2 need to be changed to:

1'. Compute an acceptable step $d^{(k)}$ of the trust-region SQP method.

2'. **if** $((QP^k_{c,\infty})$ infeasible and $\|d^{(k)}\|_\infty < \rho^k)$ **then**
fathom node **exit**
elseif $((QP^k_{c,\infty})$ infeasible and $\|d^{(k)}\|_\infty = \rho^k)$ **then**
Enter feasibility restoration phase:
repeat (SQP iteration for feasibility restoration)
(a) Compute a step of the feasibility restoration.

(b) **if** ($\|d^{(k)}\|_\infty < \rho^k$) **then** fathom node, **exit**
(c) **if** (($QP^k_{c,\infty}$) feasible) **then** return to normal SQP, **exit**
(d) **if** ($\min \|g^+(x, y)\| > 0$) **then** fathom node, **exit**
endif

The following theorem shows that Algorithm 2 converges under standard assumptions (e.g. [7] or [10]).

Theorem 1. *If f and g are smooth and convex functions, if X and Y are convex sets, if Y integer is finite and if the underlying trust-region SQP method is globally convergent, then Algorithm 2 terminates after visiting a finite number of nodes at the solution (x^*, y^*) of (P).*

Proof: The finiteness of Y integer implies that the branch-and-bound tree is finite. Thus the algorithm must terminate after visiting a finite number of nodes. Nodes are only fathomed if they are infeasible, integer feasible or if the lower bounding of Lemma 1 holds. Thus an optimal node must eventually be solved and convergence follows from the convergence of the trust-region SQP method. □

3.3. Quadratic convergence and early branching

When implementing Algorithm 2 the following adverse behaviour has been observed on some problems.

1. Some integer variables converge to a non-integral solution, namely $y_i^{(k)} \to \hat{y}_i$ but $\theta \simeq 0.02 < \tau$, i.e. the integrality gap remains bounded away from zero.
2. The SQP solver converges at second order rate during this time and $\theta \gg \|d^{(k)}\|_\infty \to 0$.

In other words the early branching heuristic is not activated during the second order convergence of the SQP method so that no early branching takes place in the final stage of the SQP solver (as $\tau = 0.1$ is too large). One way to avoid this would be to reduce τ, but this would simply repeat the problem at a smaller threshold. Choosing τ to be of the order of the tolerance of the SQP solver on the other hand would trigger early branching more often which could duplicate the work to be done if integer variables are converging to an integral solution.

These observations lead to a new early branching heuristic which takes the quadratic rate of convergence of the SQP method into account. Step 5 and 6 of Algorithm 2 are replaced by

5'. Compute the integrality gap $\theta = \max_i |y_i^{(k+1)} - \text{anint}(y_i^{(k+1)})|$
and the experimental order of convergence $p := \ln(\|d^{(k)}\|_\infty)/\ln(\|d^{(k-1)}\|_\infty)$.
6'. **if** ($\theta > \tau$ or ($p > 1.5$ and $\|d^{(k)}\|_\infty < \theta$)) **then**
Choose a non-integral $y_i^{(k+1)}$ and branch, **exit**
endif

For a quadratically convergent method one expects that $p \simeq 2$ and once quadratic convergence sets in, that $\|d^{(k+1)}\|_\infty \simeq \|d^{(k)}\|_\infty^p$. Thus if $\|d^{(k)}\|_\infty < \theta$, one cannot expect to reduce the integrality gap to zero in the remaining SQP iterations.

The new early branching rule resulted in a small but consistent reduction in the number of QP problems solved (roughly a 5% reduction) compared to the original heuristic.

3.4. Nonconvex MINLP problems

In practice many MINLP problems are nonconvex. In this case, the underestimating property of Lemma 1 no longer holds, as linearizations of nonconvex functions are not necessarily underestimators. Thus it may be possible that a node is wrongly fathomed on account of Lemma 1 and this could prevent the algorithm from finding the global minimum.

A rigorous way of extending Algorithm 2 to classes of nonconvex MINLP problems would be to replace the linearizations by linear underestimators (e.g. Horst and Tuy [20, Chapter VI]). However, it is not clear what effect this would have on the convergence properties of the SQP method. Instead the following heuristic is proposed, which consists of replacing 2' by 2":

2". **if** (($QP^k_{c,\infty}$) infeasible) **then**
  Enter feasibility restoration phase:
  repeat (SQP iteration)
    (*a*) Compute a step of the feasibility restoration.
    (*c*) **if** (($QP^k_{c,\infty}$) feasible) **then** return to normal SQP, **exit**
    (*d*) **if** ($\min \|g^+(x, y)\| > 0$) **then** fathom node, **exit**
  endif

The main difference between 2' and 2" is that a node is no longer fathomed if the QP problem is infeasible and the step lies strictly inside the trust-region ($\|d\| < \rho$). Instead a feasibility problem has to be solved to optimality (Step (*d*) above) or until a feasible QP problem is encountered (Step (*b*) above). This heuristic is motivated by the success of nonlinear branch-and-bound in solving some nonconvex MINLP problems. It ensures that a node is only fathomed if it would have been fathomed by nonlinear branch-and-bound. Clearly, there is no guarantee that the feasibility problem is solved to *global* optimality if the constraints or the objective are nonconvex.

4. Numerical experience

Nonlinear branch-and-bound and Algorithm 2 have been implemented in Fortran 77 using filter SQP [12] as the underlying NLP solver. The implementation of Algorithm 2 uses a callback routine that is invoked after each step of the SQP method. This routine manages the branch-and-bound tree stored on a stack, decides on when to branch and which problem to solve next. The use of this routine has kept the necessary changes to filterSQP to a minimum.

The test problems are from a wide range of backgrounds. Their characteristics are displayed in Table 2. The SYNTHES* problems are small process synthesis problems [7]. OPTPRLOC is a problem arising from the optimal positioning of a product in a multi-attribute space [7]. BATCH is a batch plant design problem that has been transformed to render it convex [21]. FEEDLOC is a problem arising from the optimal sizing and feed tray location of a distillation column (adapted from [27]), TRIMLOSS is a trim loss optimization problem that arises in the paper industry [29] and TRIMLON4 is an equivalent nonconvex formulation of the same problem. All problems are input as SIF files [3] plus an additional file to identify the integer variables. The test-problems are available at `www.mcs.dundee.ac.uk:8080/~sleyffer/MINLP_TP/index.html`.

Table 1 lists a description of the headers of the result tables. Table 2 list the problem characteristics. The performance of branch-and-bound and Algorithm 2 is compared in Table 3. All runs were performed on a SUN SPARCstation 4 with 128 Mb memory, using

Table 1. Description of headers of the result tables.

Header	Description
Problem	The SIF name of the problem being solved.
n	Total number of variables of the problem.
n_i	Number of integer variables of the problem.
m	The total number of constraints (excl. simple bounds).
m_n	Number of nonlinear constraints.
k	The dimension of the null-space at the solution.
#NLPs	Number of NLP problems solved (branch-and-bound).
#nodes	Number of nodes visited by Algorithm 2.
#QPs	Total number of QP (and LP) problems solved.
#f-QPs	Number of QPs in feasibility restoration.
CPU	Seconds of CPU time needed for the solve.

Table 2. Problem characteristics of test problems.

Problem	n	n_i	m	m_n	Description
SYNTHES1	6	3	6	2	Process synthesis
SYNTHES2	11	5	14	3	Process synthesis
SYNTHES3	17	8	23	4	Process synthesis
OPTPRLOC	30	25	30	25	Optimal product location
BATCH	49	24	73	1	Batch plant design
FEEDLOC	90	37	259	89	Distillation column design (nonconvex)
TRIMLOSS	142	122	75	4	Trimloss optimization
TRIMLON4	24	24	27	4	Trimloss optimization (nonconvex)

Table 3. Results for branch-and-bound and Algorithm 2.

	Branch-and-bound				Algorithm 2			
Problem	#NLPs	#QPs	(#f-QPs)	CPU	#nodes	#QPs	(#f-QPs)	CPU
SYNTHES1	5	18	(0)	0.1	5	13	(0)	0.1
SYNTHES2	17	53	(0)	0.4	17	40	(0)	0.3
SYNTHES3	25	80	(0)	0.8	25	64	(3)	0.8
OPTPRLOC	109	491	(119)	8.5	112	232	(28)	5.3
BATCH	143	371	(10)	8.9	181	391	(11)	14.8
FEEDLOC	47	340	(35)	70.8	55	103	(4)	29.0
TRIMLOSS	1336	3820	(160)	254.3	944	1624	(1)	90.3
TRIMLON4	887	1775	(66)	19.0	566	900	(112)	13.3

the compiler options `-r8 -0` and SUN's `f77` Fortran compiler. In both algorithms a depth first search with branching on the most fractional variable was implemented.

Table 3 shows that branch-and-bound only requires about 2-6 QP solves per node that is solved making it a very efficient solver. By comparison, Algorithm 2 often visits more nodes in the branch-and-bound tree. That is not surprising, as the early termination rule is based on a weaker lower bound than **FR2**. However, the overall number of QP problems that are being solved is reduced by 20%–80%. This results in a similar reduction in terms of CPU time compared to branch-and-bound. For the largest problem, Algorithm 2 is 3 times faster than nonlinear branch-and-bound.

These results are roughly in line with what one might expect taking into account that branch-and-bound solves between 3 and 5 QPs per NLP node. They are somewhat better than the results in [2].

Even though problem TRIMLON4 is nonconvex, Algorithm 2 terminated with the global minimum. For the nonconvex problem FEEDLOC, Algorithm 2 did not find an integer feasible point. This behaviour is due to the fact that the linearizations are *not* outer approximations in the nonconvex case and shows that the early termination rule does not hold for nonconvex MINLP problems. Running Algorithm 2 in nonconvex mode (see Section 3.4) produced the same minimum as branch-and-bound. As one would expect, the performance of Algorithm 2 in nonconvex mode is not as competitive as in convex mode (see Table 4) but still competitive with branch-and-bound. Better heuristics for nonconvex MINLP problems remain an open problem.

Table 4. Results for Algorithm 2 in nonconvex mode.

	Branch-and-bound				Algorithm 2			
Problem	#NLPs	#QPs	(#f-QPs)	CPU	#nodes	#QPs	(#f-QPs)	CPU
FEEDLOC	47	340	(35)	70.8	65	159	(75)	55.9
TRIMLON4	887	1775	(66)	19.0	984	1606	(259)	21.9

5. Conclusions

We have presented an algorithm for MINLP problems which interlaces the iterative solution of each NLP node with the branch-and-bound tree search. The algorithm presented here can also be interpreted as an improved lower bounding scheme for branch-and-bound. An implementation of this idea gave a factor of up to 3 improvement in terms of CPU time compared to branch-and-bound.

Acknowledgments

The author would like to thank the two referees for their very detailed and helpful comments. This work was supported by the EPSRC under grant number GR/K51204.

Note

1. Ideally one would prefer to interrupt the SQP method after each QP solve. Borchers and Mitchell use an SQP method from the NAG library which does not allow this.

References

1. B. Borchers and J.E. Mitchell, "An improved branch and bound algorithm for mixed integer nonlinear programming," Computers and Operations Research, vol. 21, no. 4, pp. 359–367, 1994.
2. B. Borchers and J.E. Mitchell, "A computational comparison of branch and bound and outer approximation methods for 0–1 mixed integer nonlinear programs," Computers and Operations Research, vol. 24, no. 8, pp. 699–701, 1997.
3. A.R. Conn, N.I.M. Gould, and Ph.L. Toint, LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A), Springer Verlag: Heidelberg, 1992.
4. A.R. Conn, N.I.M. Gould, and Ph.L. Toint, "Methods for nonlinear constraints in optimization calculations," Report RAL-TR-96-042, Rutherford Appleton Laboratory, 1996. (To appear in The State of the Art in Numerical Analysis, I. Duff and G.A. Watson (Eds.)).
5. R.J. Dakin, "A tree search algorithm for mixed integer programming problems," Computer Journal, vol. 8, pp. 250–255, 1965.
6. J.E. Dennis, M. El-Alem, and K.A. Williamson, "A trust-region approach to nonlinear systems of equalities and inequalities," SIAM Journal on Optimization, vol. 9, pp. 291–315, 1999.
7. M. Duran and I.E. Grossmann, "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," Mathematical Programming, vol. 36, pp. 307–339, 1986.
8. M. El-Hallabi and R.A. Tapia, "An inexact trust-region feasible-point algorithm for non-linear systems of equalities and inequalities," Technical Report TR-09, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, USA, 1995.
9. R. Fletcher, Practical Methods of Optimization, 2nd ed., John Wiley: Chichester, 1987.
10. R. Fletcher and S. Leyffer, "Solving mixed integer nonlinear programs by outer approximation," Mathematical Programming, vol. 66, pp. 327–349, 1994.
11. R. Fletcher and S. Leyffer, "Nonlinear programming without a penalty function," Numerical Analysis Report NA/171, Department of Mathematics, University of Dundee, Sept. 1997.
12. R. Fletcher and S. Leyffer, "User manual for filter SQP," Numerical Analysis Report NA/181, Dundee University, April 1998.
13. O.E. Flippo and A.H.G. Rinnoy Kan, "Decomposition in general mathematical programming," Mathematical Programming, vol. 60, pp. 361–382, 1993.

14. C.A. Floudas, Nonlinear and Mixed-Integer Optimization, Oxford University Press: New York, 1995.
15. A.M. Geoffrion, "Generalized benders decomposition," Journal of Optimization Theory and Applications, vol. 10, pp. 237–260, 1972.
16. I.E. Grossmann and Z. Kravanja, "Mixed-integer nonlinear programming: A survey of algorithms and applications," in Large-Scale Optimization with Applications, Part II: Optimal Design and Control, A.R. Conn, L.T. Biegler, T.F. Coleman, and F.N. Santosa (Eds.), Springer: New York, Berlin, 1997.
17. I.E. Grossmann and R.W.H. Sargent, "Optimal design of multipurpose batch plants," Ind. Engng. Chem. Process Des. Dev., vol. 18, pp. 343–348, 1979.
18. O.K. Gupta and A. Ravindran, "Branch and bound experiments in convex nonlinear integer programming," Management Science, vol. 31, pp. 1533–1546, 1985.
19. S.P. Han, "A globally convergent method for nonlinear programming," Journal of Optimization Theory and Applications, vol. 22, no. 3, pp. 297–309, 1977.
20. R. Horst and H. Tuy, Global Optimization: Deterministic Approaches, 2nd ed., Springer-Verlag: Berlin, 1993.
21. G.R. Kocis and I.E. Grossmann, "Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis," Industrial Engineering Chemistry Research, vol. 27, pp. 1407–1421, 1988.
22. A.H. Land and A.G. Doig, "An automatic method for solving discrete programming problems," Econometrica, vol. 28, pp. 497–520, 1960.
23. M.J.D. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in Numerical Analysis, 1977, G.A. Watson (Ed.), Springer Verlag: Berlin 1978, pp. 144–157.
24. I. Quesada and I.E. Grossmann, "An LP/NLP based branch-and-bound algorithm for convex MINLP optimization problems," Computers and Chemical Engineering, vol. 16, pp. 937–947, 1992.
25. A.J. Quist, R. van Geemert, J.E. Hoogenboom, T. Illés, E. de Klerk, C. Roos, and T. Terlaky, "Optimization of a nuclear reactor core reload pattern using nonlinear optimization and search heuristics," Draft paper, Delft University of Technology, Faculty of Applied Mathematics, Department of Operation Research, Mekelweg 4, 2628 CD Delft, The Netherlands, Sept. 1997.
26. R.A. Stubbs and S. Mehrotra, "A branch-and-cut method for 0–1 mixed convex programming," Technical report 96-01, Department of IE/MS, Northwestern University, Evanston, IL 60208, USA, Jan. 1996.
27. J. Viswanathan and I.E. Grossmann, "Optimal feed location and number of trays for distillation columns with multiple feeds," I&EC Research, vol. 32, pp. 2942–2949, 1993.
28. O.V. Volkovich, V.A. Roshchin, and I.V. Sergienko, "Models and methods of solution of quadratic integer programming problems," Cybernetics, vol. 23, pp. 289–305, 1987.
29. T. Westerlund, J. Isaksson, and I. Harjunkoski, "Solving a production optimization problem in the paper industry," Report 95-146-A, Department of Chemical Engineering, Abo Akademi, Abo, Finland, 1995.
30. R.B. Wilson, "A simplicial algorithm for concave programming, Ph.D. Thesis, Harvard University Graduate School of Business Administration, 1963.
31. Y. Yuan, "Trust region algorithms for nonlinear equations," Technical Report 049, Department of Mathematics, Hong Kong University, Hong Kong, 1994.