GENERALIZED OUTER APPROXIMATION

Introduction.

This article deals with the solution of *Mixed Integer Nonlinear Programming* (MINLP) problems of the form

$$\mathrm{P}\left\{\begin{array}{ll}\min\limits_{x,y} & f(x,y)\\ \text{subj.to} & g(x,y)\le 0\\ & x\in X,\ y\in Y \text{ integer}\ .\end{array}\right.$$

Throughout the following general assumptions are made: *(A1)* f and g are twice continuously differentiable and convex functions,
(A2) X and Y are nonempty compact convex (polyhedral) sets and
(A3) a constraint qualification holds at the solution of every NLP subproblem obtained by fixing the integer variables y.

MINLP problems arise in a range of engineering applications (see e.g., Floudas [8] and Grossmann and Kravanja [10] and references therein).

A class of methods for MINLP problems is discussed, which provide an alternative to nonlinear branch and bound [3]. These algorithms are based on the concept of defining an *MILP master problem*. Relaxations of such a master problem are then used in constructing algorithms for solving the MINLP problem.

The methods presented here are a generalization of Outer Approximation proposed by Duran and Grossmann [4] (see also Yuan et.al. [14]) and of LP/NLP based branch and bound of Quesada and Grossmann [13].

The next section presents the reformulation of P as an MILP master program. Based on this reformulation two algorithms are presented in the following sections which solve a finite sequence of NLP subproblems and MILP or *MIQP master problems*, respectively. The final section shows how the re–solution of these master problems can be avoided by updating their branch and bound trees.

Outer Approximation of P.

In this section the MINLP model problem P is reformulated as an MILP problem using Outer Approximation. The reformulation employs projection onto the integer variables and linearization of the resulting NLP subproblems by means of supporting hyperplanes. The convexity assumption allows an MILP formulation to be given where all supporting hyperplanes are collected in a single MILP.

In order to improve the readability of the material, the reformulation is first done under the simplifying assumption that all integer assignments $y\in Y$ are feasible. Next a rigorous treatment of infeasible subproblems is outlined, correcting an inaccuracy in [4] and [14], which could cause the algorithm to cycle. Finally, the two parts are combined and the correctly reformulated MILP master program is presented.

The reformulation presented in the next section affords new insight into Outer Approximation. It can be seen, for example, that it suffices to add the linearizations of *strongly active* constraints to the master program. This is very important since it reduces the size of the MILP master program relaxations that are solved in the Outer Approximation Algorithms.

When all $y\in Y$ are feasible.

In this subsection the simplifying assumption is made that all $y\in Y$ are feasible. The first step in reformulating P is to define the *NLP subproblem*

$$\mathrm{NLP}(y^j)\left\{\begin{array}{ll}\min\limits_{x} & f(x,y^j)\\ \text{subj.to} & g(x,y^j)\le 0\\ & x\in X\end{array}\right.$$

in which the integer variables are fixed at the value $y=y^j$. By defining $v(y^j)$ as the optimal value of the subproblem $\mathrm{NLP}(y^j)$ it is possible to express P in terms of a projection on to the y variables, that is

$$\min_{y^j\in Y}\left(v(y^j)\right). \tag{1}$$

The assumption that all $y\in Y$ are feasible implies that *all* subproblems are feasible. Let x^j denote an optimal solution of $\mathrm{NLP}(y^j)$ for $y^j\in Y$ (existence of x^j follows by the compactness of X). Because a constraint qualification holds at the solution of every subproblem $\mathrm{NLP}(y^j)$ for every $y^j\in Y$, it follows that (1) has the same optimal value as the problem

$$\min_{y^j\in Y}\left(u(y^j)\right) \tag{2}$$

where $u(y^j)$ is the optimal value of the following LP

$$\left\{\begin{array}{ll}\min\limits_{x} & f^j + (\nabla f^j)^T \begin{pmatrix} x - x^j \\ 0 \end{pmatrix} \\ \text{subj.to} & 0 \geq g^j + [\nabla g^j]^T \begin{pmatrix} x - x^j \\ 0 \end{pmatrix} \\ & x \in X. \end{array}\right. \quad (3)$$

In fact it suffices to include those linearizations of constraints about (x^j, y^j) which are strongly active at the solution of the corresponding subproblem. Here $f^j = f(x^j, y^j)$ and $\nabla f^j = \nabla f(x^j, y^j)$ etc.

It is convenient to introduce a dummy variable $\eta \in \mathbb{R}$ into (3), giving rise to the equivalent problem

$$\left\{\begin{array}{ll}\min\limits_{x,\eta} & \eta \\ \text{subj.to} & \eta \geq f^j + (\nabla f^j)^T \begin{pmatrix} x - x^j \\ 0 \end{pmatrix} \\ & 0 \geq g^j + [\nabla g^j]^T \begin{pmatrix} x - x^j \\ 0 \end{pmatrix} \\ & x \in X. \end{array}\right.$$

The convexity assumption *(A1)* implies that (x^i, y^i) is feasible in the inner optimization problem above for all $y^i \in Y$, where x^i is an optimal solution to NLP(y^i). Thus an equivalent MILP problem

$$\mathrm{M}_Y \left\{\begin{array}{ll}\min\limits_{x,y,\eta} & \eta \\ \text{subj.to} & \eta \geq f^j + (\nabla f^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & 0 \geq g^j + [\nabla g^j]^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & \forall y^j \in Y \\ & x \in X,\ y \in Y \text{ integer} \end{array}\right.$$

is obtained. That is M_Y has one set of linearizations of the objective and constraint functions per integer point $y^j \in Y$.

Infeasible subproblems.

Usually, not all $y \in Y$ give rise to feasible subproblems. Defining the sets

$$T = \left\{ j : x^j \text{optimal solution to NLP}(y^j) \right\}$$

$$\mathrm{V} = \{ y \in Y : \exists\, x \in X \text{ with } g(x, y) \leq 0 \}.$$

Then V is the set of all integer assignments y that give rise to feasible NLP subproblems and T is the set of indices of these integer variables. Then P can be expressed as a projection on to the integer variables.

$$\min_{y^j \in V} \left(v(y^j) \right).$$

In this projection the set V replaces Y in (1). The equivalent MILP problem is now given by

$$\mathrm{M}_V \left\{\begin{array}{ll}\min\limits_{x,y,\eta} & \eta \\ \text{subj.to} & \eta \geq f^j + (\nabla f^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & 0 \geq g^j + [\nabla g^j]^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & \forall j \in T \\ & x \in X,\ y \in V \text{ integer} \end{array}\right.$$

obtained from M_Y by replacing Y by V.

It remains to find a suitable representation of the constraint $y \in V$ by means of supporting hyperplanes. The master problem given in [4] is obtained from problem M_V by replacing $y \in V$ by $y \in Y$. Duran and Grossmann (1986) justify this step by arguing that a representation of the constraints $y \in V$ is included in the linearizations in problem M_V. However, it is not difficult to derive an MINLP problem where this would lead to an incorrect master problem [6], [11, p. 79].

In order to derive a correct representation of $y \in V$ it is necessary to consider how NLP solvers detect infeasibility. Infeasibility is detected when convergence to an optimal solution of a feasibility problem occurs. At such an optimum, some of the nonlinear constraints will be violated and other will be satisfied and the norm of the infeasible constraints can only be reduced by making some feasible constraints infeasible. A suitable framework for *nonlinear feasibility problems* in the context of Outer Approximation is

$$F(y^k) \left\{\begin{array}{ll}\min\limits_{x} & \sum\limits_{i \in J^\perp} w_i^k g_i^+(x, y^k) \\ \text{subj.to} & g_j(x, y^k) \leq 0,\ j \in J \\ & x \in X \end{array}\right.$$

The constraints in F(y^k) have been divided into two sets: one that can be satisfied and another that cannot be satisfied. Infeasible subproblems now correspond to solutions of F(y^k) with $\sum_{i \in J^\perp} w_i^k g_i^+(x, y^k) > 0$. Most common feasibility problems such as l_1 and l_∞ as well as the

feasibility filter [7] fit into this framework. The following lemma shows how solutions of $\mathrm{F}(y^k)$ can be used to construct a representation of $y \in V$.

Lemma 1 *If NLP(y^k) is infeasible, so that x^k solves F(y^k) with*

$$\sum_{i \in J^\perp} w_i^k (g_i^k)^+ > 0 \tag{4}$$

then $y = y^k$ is infeasible in the constraints

$$0 \geq g_i^k + (\nabla g_i^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \quad \forall i \in J^\perp$$
$$0 \geq g_j^k + (\nabla g_j^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \quad \forall j \in J,$$

for all $x \in X$.

The proof of this Lemma can be found in [6, Lemma 1].

The general case.

This subsection completes the derivation of the MILP master program by combining the developments of the previous two subsections. Let the integer assignment y^k produce an infeasible subproblem and denote

$$S = \left\{ k : \mathrm{NLP}(y^k) \text{ infeasible, } x^k \text{ solves } \mathrm{F}(y^k) \right\}.$$

Note that S is the complement of the set T defined in the previous subsection. It then follows directly from Lemma 1 that the constraints

$$0 \geq g^k + [\nabla g^k]^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \; \forall k \in S$$

exclude all integer assignments y^k for which $\mathrm{NLP}(y^k)$ is infeasible. Thus a general way to correctly represent the constraints $y \in V$ in M_V is to add linearizations from $\mathrm{F}(y^k)$ when infeasible subproblems are obtained, giving rise to the following MILP master problem.

$$\mathrm{M} \left\{ \begin{array}{ll} \min\limits_{x,y,\eta} & \eta \\ \text{subj.to} & \eta \geq f^j + (\nabla f^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & 0 \geq g^j + [\nabla g^j]^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & \forall j \in T \\ & 0 \geq g^k + [\nabla g^k]^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \\ & \forall k \in S \\ & x \in X,\ y \in Y \text{ integer.} \end{array} \right.$$

The development of the preceding two subsections provides a proof of the following result:

Theorem 2 *If assumptions (A1), (A2) and (A3) hold, then M is equivalent to P in the sense that (x^*, y^*) solves P if and only if it solves M.*

Problem M is an MILP problem, but it is not practical to solve M directly, since this would require all subproblems $\mathrm{NLP}(y^j)$ to be solved first. This would be a very inefficient way of solving problem P. Therefore, instead of attempting to solve M directly, relaxations of M are used in an iterative process that is the subject of the next section.

Linear Outer Approximation.

This section describes, how relaxations of the master program M, developed in the previous section can be employed to solve the model problem P. The resulting algorithm is termed *linear outer approximation*. It is shown to iterate finitely between NLP subproblems and MILP master program relaxations. This algorithm is seen to be less efficient if curvature information is present in the problem. A worst case example, in which linear outer approximation visits all integer assignments has been derived in [6]. This example motivates the introduction of a second order term into the MILP master program relaxations, resulting in a *quadratic Outer Approximation* algorithm which is considered in the next section.

Each iteration of the linear outer approximation algorithm chooses a new integer assignment y^i and attempts to solve $\mathrm{NLP}(y^i)$. Either a feasible solution x^i is obtained or infeasibility is detected and x^i is the solution of a feasibility problem $\mathrm{F}(y^i)$ (other pathological cases are eliminated by the assumption that the set X is compact). The algorithm replaces the sets T and S in M by the sets

$$T^i = \left\{ j \mid j \leq i : x^j \text{ solution to } \mathrm{NLP}(y^j) \right\}$$
$$S^i = \left\{ k \mid k \leq i : x^k \text{ solution to } F(y^k) \right\}.$$

It is also necessary to prevent any $y^j,\ j \in T^i$ from becoming the solution of the relaxed master problem. This can be done by including a constraint

$$\eta < \mathrm{UBD}^i$$

where

$$\text{UBD}^i = \min\{f^j : \ \ j \in T^i\}$$

is an upper bound on the optimum. Thus the following master problem is defined

$$M^i \left\{ \begin{array}{ll} \min\limits_{x,y,\eta} & \eta \\ \text{subj.to} & \eta < \text{UBD}^i \\ & \eta \geq f^j + \nabla(f^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & 0 \geq g^j + \nabla[g^j]^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & \forall j \in T^i \\ & 0 \geq g^k + \nabla[g^k]^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \\ & \forall k \in S^i \\ & x \in X,\ y \in Y \text{ integer.} \end{array} \right.$$

The algorithm solves M^i to obtain a new integer assignment y^{i+1}, and the whole process is repeated iteratively. A detailed description of the algorithm is as follows.

Algorithm 1: *Linear Outer Approximation*
Initialization: y^0 given; set $i = 0$, $T^{-1} = \emptyset$, $S^{-1} = \emptyset$ and $\text{UBD}^{-1} = \infty$.
REPEAT

1. Solve NLP(y^i) or F(y^i) as appropriate. Let the solution be x^i.
2. Linearize objective and constraint functions about (x^i, y^i). Set $T^i = T^{i-1} \cup \{i\}$ or $S^i = S^{i-1} \cup \{i\}$ as appropriate.
3. IF (NLP(y^i) feasible & f^i <UBD^{i-1}) THEN
   update current best point by setting
   $x^* = x^i$, $y^* = y^i$, $\text{UBD}^i = f^i$.
   ELSE
   Set $\text{UBD}^i = \text{UBD}^{i-1}$.
4. Solve the current relaxation M^i of the master program M, giving a new y^{i+1}. Set $i = i + 1$.

UNTIL (M^i is infeasible).

The figure below illustrates the different stages of the algorithm.

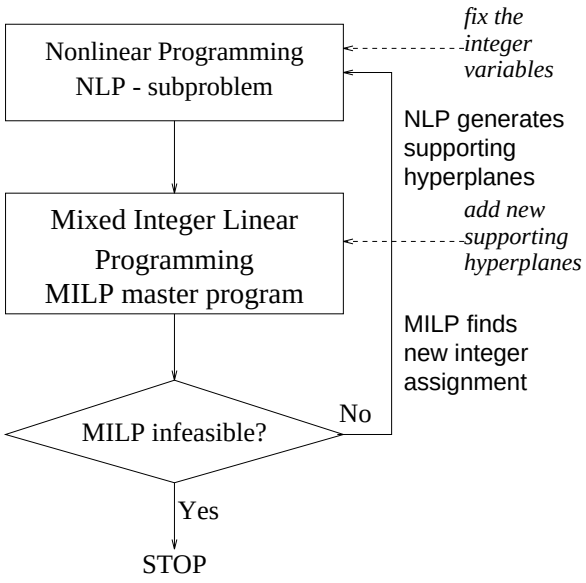


Illustration of Linear Outer Approximation

The algorithm also detects whether or not P is infeasible. If UBD= ∞ on exit then all integer assignments that are visited by the algorithm are infeasible (i.e. Step 3 is not invoked). The use of upper bounds on η and the definition of the sets T^i and S^i ensure that no y^i is replicated by the algorithm. This enables a proof to be given that the algorithm terminates after a finite number of steps, provided that there is only a finite number of integer assignments.

Theorem 3 *If assumptions (A1), (A2) and (A3) hold, and if $|Y| < \infty$, then Algorithm 1 terminates in a finite number of steps at an optimal solution of P or with an indication that P is infeasible.*

A proof of Theorem 3 can be found in [6]. Below a brief outline of the proof is given: The optimality of x^i in NLP(y^i) implies that $\eta \geq f^i$ for any feasible point in (3). The upper bound $\eta < f^i$ therefore ensures that the choice $y = y^i$ in M^k has no feasible points $x \in X$. Therefore the algorithm is finite. The optimality of the algorithm follows from the convexity of f and g which ensures that the linearizations are supporting hyperplanes.

Quadratic Outer Approximation.

Curvature can often play an important role in optimization. If this is the case, then an algorithm based on *linear* representations of the problem functions can be inefficient. In [6], a

worst case example is given for which linear outer approximation visits all integer points. This motivates the introduction of a curvature information into the master programs. In the remainder of this section it is shown how this can be achieved for linear outer approximation by including a second order Lagrangian term into the objective function of the MILP master programs.

These considerations have led to the development of a new algorithm based on the use of second order information. The development of such an algorithm seems contradictory at first sight, since quadratic functions do not provide underestimators of general convex functions. However, the derivation of the previous section allows the inclusion of a curvature term into the objective function of the MILP master problem. This quadratic term influences the choice of the next iterate by the algorithm without surrendering the finite convergence properties which rely on the fact that the feasible region of the master problem is an outer approximation of the feasible region of the MINLP problem P. The resulting algorithm is referred to as *quadratic Outer Approximation* and is obtained by replacing the relaxed master problem M^i by the MIQP problem Q^i in Step 4 of Algorithm 1.

The new master problem Q^i can be defined as

$$Q^i \left\{ \begin{array}{ll} \min\limits_{x,y,\eta} & \eta + q^i(x,y) \\ \text{subj.to} & \eta < \text{UBD}^i \\ & \eta \geq f^j + (\nabla f^j)^T \left(\begin{array}{c} x - x^j \\ y - y^j \end{array} \right) \\ & 0 \geq g^j + [\nabla g^j]^T \left(\begin{array}{c} x - x^j \\ y - y^j \end{array} \right) \\ & \forall j \in T^i \\ & 0 \geq g^k + [\nabla g^k]^T \left(\begin{array}{c} x - x^k \\ y - y^k \end{array} \right) \\ & \forall k \in S^i \\ & x \in X,\ y \in Y \text{ integer} . \end{array} \right.$$

where the *quadratic* term $q^i(x,y)$ is defined by

$$q^i(x,y) = \frac{1}{2} \left(\begin{array}{c} x - x^i \\ y - y^i \end{array} \right)^T \nabla^2 \mathcal{L}^i \left(\begin{array}{c} x - x^i \\ y - y^i \end{array} \right)$$

and

$$\mathcal{L}^i = \mathcal{L}(x^i, y^i, \lambda^i) = f(x^i, y^i) + (\lambda^i)^T g(x^i, y^i)$$

is the usual Lagrangian function.

Numerical experience in [11, Chapter 6] indicates that adding a curvature term reduces the number of iterations of Outer Approximation if general integer variables are present. However, the iteration count is not reduced for problems involving binary variables only. As a consequence these preliminary results indicate that quadratic Outer Approximation only improves the computation times for problems with general integer variables, as MIQP problems are usually more expensive to solve than MILP problems.

Avoiding re–solving the master problems.

This section presents a new approach to the solution of successive master program relaxations. It has been proposed by Quesada and Grossmann [13] for a class of problems whose objective and constraint functions are linear in the integer variables and is called *LP/NLP based branch and bound* algorithm. Their approach is generalized here to cover problems with nonlinearities in the integer variables. In addition a new algorithm *QP/NLP based branch and bound* is proposed based on the quadratic master program Q^i which takes curvature information into account.

The motivation for the LP/NLP based branch and bound algorithm is that outer approximation usually spends an increasing amount of computing time in solving successive MILP master program relaxations. Since the MILP relaxations are strongly related to one another this means that a considerable amount of information is re-generated each time a relaxation is solved. The new approach avoids the re–solution of MILP master program relaxations by updating the branch and bound tree. This section makes extensive use of branch and bound terminology and the reader is referred to the extensive literature on branch and bound (e.g., [1], [2], [8], [9], [12]) for the relevant definitions.

Instead of solving successive relaxations of M, the new algorithm solves only one MILP problem which is updated as new integer assignments are encountered *during* the tree search. Initially an NLP–subproblem is solved and the initial master program relaxation M^0 is set up from the supporting hyperplanes at the solution of

the NLP–subproblem. The MILP problem M^0 is then solved by a branch and bound process with the exception that each time a node (corresponding to an LP problem) gives an integer feasible solution y^i, say, the process is interrupted and the corresponding NLP(y^i) subproblem is solved. New linearizations from NLP(y^i) are then added to every node on the stack, effectively updating the branch and bound tree. The branch and bound process continues in this fashion until no problems remain on the stack. At that moment all nodes are fathomed and the tree search is exhausted.

Unlike ordinary branch and bound a node cannot be assumed to have been fathomed, if it produces an integer feasible solution, since the previous solution at this node is cut out by the linearizations added to the master program. Thus only infeasible nodes can be assumed to be fathomed. In the case of MILP master programs there exists an additional opportunity for pruning. Since the LP nodes are outer approximations of the MINLP subproblem corresponding to their respective subtree a node can be regarded as fathomed if its lower bound is greater than or equal to the current upper bound UBD^i.

Algorithm 2: *LP/NLP based branch and bound*
Initialization: y^0 given; set $i = 1$, $T^{-1} = \emptyset$, $S^{-1} = \emptyset$.
Set up the initial master program:
• Solve NLP(y^0). Let the solution be x^0.
• Linearize objective and constraint functions about (x^0, y^0). Set $T^0 = \{0\}$.
• Set $x^* = x^0$, $y^* = y^0$, UBD$^0 = f^0$.
Place M^0 with its integer restrictions relaxed on the stack.
WHILE (stack is not empty) DO BEGIN

1. Remove a problem (P') from the stack and solve the LP giving (x', y', η'). η' is a lower bound for all NLP child problems whose root is the current problem.
2. IF (y' integer) THEN
   • Set $y^i = y'$ & solve NLP(y^i) or F(y^i). Let the solution be x^i.
   • Linearize objective and constraint functions about (x^i, y^i). Set $T^i = T^{i-1} \cup \{i\}$ or $S^i = S^{i-1} \cup \{i\}$.
   • Add linearizations to *all* pending problems on the stack.

   IF (NLP(y^i) feasible & $f^i <$UBD^i)
   • Update best point $x^* = x^i$, $y^* = y^i$ UBD$^{i+1} = f^i$.

   ELSE
   • Set UBD^{i+1} = UBD^i.

   ENDIF
   • Place (P') back on stack; set $i = i + 1$.
   • *Pruning*: Remove all problems from stack with $\eta' >$UBD^{i+1}.

   ELSE
   • Branch on an integer variable and add two new problems to the stack.

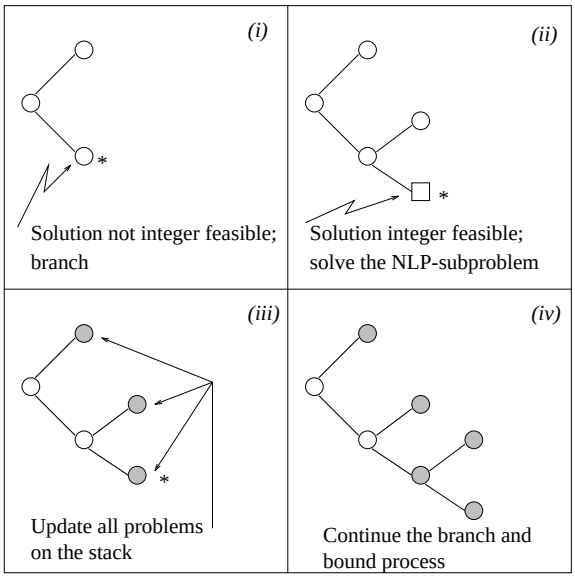
   ENDIF

END (WHILE–LOOP)

As in the two outer approximation algorithms the use of an upper bound implies that no integer assignment is generated twice during the tree search. Since both the tree and the set of integer variables are finite the algorithm eventually encounters only infeasible problems and the stack is thus emptied so that the procedure stops. This provides a proof of the following corollary to Theorem 3.

Corollary 4 *If assumptions (A1), (A2) and (A3) hold, and if $|Y| < \infty$, then Algorithm 2 terminates in a finite number of steps at a solution of P or with an indication that P is infeasible.*

The figure below illustrates the progress of Algorithm 2 In *(i)*, the LP relaxation of the initial MILP has been solved and two branches added to the tree. The LP that is solved next (indicated by an *) does not give an integer feasible solution and two new branches are introduced. The next LP in *(ii)* produces an integer feasible solution indicated by a box. The corresponding NLP subproblem is solved and in *(iii)* all nodes on the stack are updated (indicated by the shaded circles) by adding the linearizations from the NLP subproblem including the upper bound UBD^i which cuts out the current assignment y^i. Then, the branch and bound process continues on the updated tree by solving the LP marked by a *.



Progress of LP/NLP based branch and bound

If curvature information plays an important part in the problem P, then it may be beneficial to add a quadratic term $q^i(x, y)$ to the master problem. This gives rise to *QP/NLP based branch and bound* algorithm. It differs from Algorithm 2 in two important aspects. The first difference is that QP rather than LP problems are solved in the tree search. The second difference is a consequence of the first. Since QP problems do not provide lower bounds on the MINLP problems P, the pruning step in Algorithm 2 cannot be applied.

In preliminary numerical experiments in [11, Chapter 6] and [5] it has been observed that the LP and QP version of Algorithm 2 are usually faster than their counterparts based on Algorithm 1, often beating the latter by a factor of 2. A detailed numerical comparison of the two approaches is still outstanding.

References

[1] Beale, E.M.L.: 'Integer Programming', in D.A.H. Jacobs (ed.): *The State of the Art in Numerical Analysis*, Academic Press, 1978.

[2] Borchers, B., and Mitchell, J.E.: 'An improved branch and bound algorithm for Mixed Integer Nonlinear Programming', *Computers and Operations Research* **21**, no. 4 (1994), 359–367.

[3] Dakin, R.J.: 'A tree search algorithm for mixed integer programming problems', *Computer Journal* **8** (1965), 250–255.

[4] Duran, M., and Grossmann, I.E.: 'An outer-approximation algorithm for a class of Mixed–Integer Nonlinear Programs', *Mathematical Programming* **36** (1986), 307–339.

[5] Fletcher, R., and Leyffer, S.: Computing Lower Bounds for MIQP Branch–and–Bound, Numerical Analysis Report NA/151, University of Dundee, Department of Mathematics and Computer Science, Dundee, Scotland, UK, June 1994, To appear in SIAM Journal on Optimization.

[6] Fletcher, R., and Leyffer, S.: 'Solving mixed integer nonlinear programs by outer approximation', *Mathematical Programming* **66** (1994), 327–349.

[7] Fletcher, R., and Leyffer, S.: Nonlinear Programming without a penalty function, Numerical Analysis Report NA/171, Department of Mathematics, University of Dundee, Sept. 1997.

[8] Floudas, C.A.: *Nonlinear and Mixed–Integer Optimization*, Topics in Chemical Engineering. Oxford University Press, New York, 1995.

[9] Garfinkel, R.S., and Nemhauser, G.L.: *Integer Programming*, John Wiley, New York, 1972.

[10] Grossmann, I.E., and Kravanja, Z.: 'Mixed–Integer Nonlinear Programming: A Survey of Algorithms and Applications', in A.R. Conn L.T. Biegler, T.F. Coleman and F.N. Santosa (eds.): *Large–Scale Optimization with Applications, Part II: Optimal Design and Control*, Springer, 1997.

[11] Leyffer, S.: *Deterministic Methods for Mixed Integer Nonlinear Programming*, PhD thesis, University of Dundee, Dundee, Scotland, UK, Dec. 1993.

[12] Nemhauser, G.L., and Wolsey, L.A.: *Integer and Combinatorial OPtimization*, John Wiley, New York, 1988.

[13] Quesada, I., and Grossmann, I.E.: 'An LP/NLP based branch–and–bound algorithm for convex MINLP optimization problems', *Computers and Chemical Engineering* **16** (1992), 937–947.

[14] X. Yuan, S. Zhang, L. Pibouleau, and Domenech, S.: 'Une méthode d'optimization non linéaire en variables mixtes pour la conception de procédés', *Operations Research*, no. 22 (1988), 331–346.

S. Leyffer
Department of Mathematics
University of Dundee
Dundee
Scotland, UK
E-mail address: sleyffer@mcs.dundee.ac.uk

AMS 1991 Subject Classification: 90C11, 90C30, 49M20.
Key words and phrases: Mixed Integer Nonlinear Programming, Outer Approximation, Branch and Bound.