Solving Mixed Integer Nonlinear Programs by Outer Approximation

Roger Fletcher and Sven Leyffer*
University of Dundee

October 16, 1996

Abstract

A wide range of optimization problems arising from engineering applications can be formulated as Mixed Integer NonLinear Programmming problems (MINLPs). Duran and Grossmann (1986) suggest an outer approximation scheme for solving a class of MINLPs that are linear in the integer variables by a finite sequence of relaxed MILP master programs and NLP subproblems.

Their idea is generalized by treating nonlinearities in the integer variables directly, which allows a much wider class of problem to be tackled, including the case of pure INLPs. A new and more simple proof of finite termination is given and a rigorous treatment of infeasible NLP subproblems is presented which includes all the common methods for resolving infeasibility in Phase I.

The worst case performance of the outer approximation algorithm is investigated and an example is given for which it visits all integer assignments. This behaviour leads us to include curvature information into the relaxed MILP master problem, giving rise to a new quadratic outer approximation algorithm.

An alternative approach is considered to the difficulties caused by infeasibility in outer approximation, in which exact penalty functions are used to solve the NLP subproblems. It is possible to develop the theory in an elegant way for a large class of nonsmooth MINLPs based on the use of convex composite functions and subdifferentials, although an interpretation for the l_1 norm is also given.

Key words: Nonlinear integer programming, mixed integer nonlinear programming, decomposition, outer approximation.

1 Introduction

Many optimization problems involve *integer* or *discrete variables* and can be modelled as Mixed Integer Nonlinear Programming problems (MINLPs). These variables can variously be integer variables modelling for example numbers of men, or zero–one variables modelling decisions, or discrete variables modelling, for example, equipment sizes. In addition to these there may also exist *continuous variables* which for example may represent pressures or temperatures. Nonlinearities come into the model either through physical properties such as enthalpy and vapour/liquid equilibrium, or may also involve the decision variables through for example economies of scale. The function to be optimized in this context usually involves costs and profits from the design.

*This work is supported by SERC grant no SERC GR/F 07972

Throughout the paper no distinction will be made between the different types of integer or discrete variables referred to above. This does not constitute a great loss of generality if it is assumed that the underlying MILP or MIQP solver is capable of handling any type of discrete variable. It is also possible to replace a discrete variable by an integer variable or by a number of zero–one variables. In the rest of the paper the term integer variables is taken to include the possibility of discrete but non–integer variables.

The class of problem to be considered here is

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

and the following assumptions are made

A1 X is a nonempty compact convex set defined by a system of linear inequality constraints and the functions

$$\begin{aligned} f &: \mathbb{R}^n\times\mathbb{R}^p \rightarrow \mathbb{R}\\ g &: \mathbb{R}^n\times\mathbb{R}^p \rightarrow \mathbb{R}^m\end{aligned}$$

are convex.

A2 f and g are once continuously differentiable.

A3 A constraint qualification holds at the solution of every NLP subproblem which is obtained from P by fixing the integer variables y. This could be, for instance, the assumption that the set of feasible directions at the solution can be identified with the set of feasible directions for the constraint linearized at the solution (e. g. [3], p. 202), for which a sufficient condition is that the normal vectors of the active constraints are linearly independent.

Assumptions **A1** to **A3** are not unreasonable in the sense that both Duran and Grossmann's outer approximation algorithm [2] and the nonlinear branch and bound algorithm require equivalent assumptions to hold, to ensure that P and all subproblems or relaxations of P are solved correctly.

The most serious restriction is that of convexity of f and g, and often the convexity assumptions do not hold in practice. The proposed algorithm can be applied to nonconvex problems, although no guarantee can be given that the solution obtained by the algorithm is a global solution. Kocis and Grossmann [6] give heuristics which aim to solve nonconvex MINLP and these are also applicable to the algorithms presented here. The model problem P is otherwise of general applicability and allows nonlinearities in the integer variables to be treated directly, thus including the case of pure INLP problems in the class of treatable problems. In [2] Duran and Grossmann proposed an outer approximation algorithm for solving the subclass of P where f and g are linear in the integer variables. Their algorithm has proved very successful in practice and its merits are examined in a more general context in this paper.

This outer approximation routine solves P by alternating finitely between an NLP subproblem (obtained from P by fixing the integer variables y) and relaxations of an MILP master program. Alternatively P may be solved using Generalized Benders Decomposition as proposed by Geoffrion in [5] and generalized to cover problems with nonlinear terms in the integer variables y by Flippo et. al. [4]. Like outer approximation Generalized Benders Decomposition solves P by alternating finitely between an NLP subproblem and an MILP master problem. The difference lies in the derivation of the MILP master program, and in Generalized Benders Decomposition nonlinear duality theory rather than

outer approximation is employed to obtain the MILP master problem. As has been pointed out in [2] the lower bounds produced by solving the master program relaxation of outer approximation are tighter than the lower bounds produced by Generalized Benders Decomposition. However, since Generalized Benders Decomposition adds less constraints to the master program at each iteration no conclusions as to the final efficiency can be derived from this property. Finally there is the possibility to use an NLP based branch and bound algorithm, where at each node of the tree an NLP has to be solved and branches are added in form of additional bounds on the integer variables.

In the remainder of this section a brief outline of the paper is given. In Section 2 a general Phase I feasibility problem is posed and it is described how this problem can be used to generate cuts that exclude integer assignments y that give rise to infeasible subproblems. The section is self–contained and is applicable to various NLP solvers. Section 3 shows how P can be reformulated as an equivalent MILP problem using projection onto the space of integer variables together with outer approximation by supporting hyperplanes. The result of Section 2 enables a rigorous treatment of infeasible NLP subproblems to be given. In Section 4 a new outer approximation algorithm is developed based on the reformulation of Section 3. The algorithm iterates finitely between NLP subproblems and MILP master problem relaxations. A new and more simple proof of the finite convergence property is given. The worst case performance of the algorithm is studied and an example is provided which shows that the algorithm can be very inefficient if nonlinearities are present. This motivates us to investigate algorithms which solve MIQP master problems in an attempt to to take second order information into account. In Section 5 an outer–approximation algorithm is developed for a class of nonsmooth MINLPs. An important example for a nonsmooth MINLP is given by an exact penalty function formulation of P. This offers an alternative approach to the difficulties caused by infeasible subproblems. The theory is developed in a general setting based on the use of convex composite functions and subdifferentials, although an interpretation for the l_1 norm is also given. Conditions under which the penalty function formulation is equivalent to P are given and the outer–approximations are compared with the corresponding Benders cuts.

Throughout this paper the superscript ^{i} is used to indicate the iteration count, superscript ^{j} is used to index the feasible NLP subproblems defined in Section 3 and superscript ^{k} to index infeasible subproblems. The following notation is adopted to distinguish between function values and functions. $f^j = f(x^j, y^j)$ denotes the value of f evaluated at the point (x^j, y^j), similarly $\nabla f^j = \nabla f(x^j, y^j)$ is the value of the gradient of f at (x^j, y^j). The same conventions apply for all other functions and higher derivatives.

2 Infeasibility in NLP problems

NLP solvers detect infeasibility in an NLP problem in many ways. Consider the constraints of an NLP problem.

$$\begin{cases} g_i(x) \leq 0, & i = 1, 2, \ldots, m \\ x \in X \subset \mathbb{R}^n \end{cases}$$

where the set X might for example be simple upper and lower bounds on x. Two obvious possibilities for finding a feasible point are to attempt to minimize an l_1 or l_∞ sum of constraint violations, that is

$$\min_x \sum_{i=1}^{m} g_i^+(x), \; x \in X$$

or

$$\min_x \max_{i=1,\ldots,m} g_i^+(x),\ x \in X$$

where $a^+ = \max(a, 0)$. Other methods aim to maintain feasibility in any constraint residual once it has been established. For example an l_1 problem of the form

$$\begin{cases} \min\limits_x & \sum\limits_{i \in J^\perp} g_i^+(x) \\ \text{subject to} & g_j(x) \le 0,\ j \in J \\ & x \in X \end{cases}$$

might be solved, in which the set J is the set of constraints which are currently feasible in the algorithm and $J^\perp$ is its complement. Alternatively constraints may be driven to feasibility one at a time, whilst maintaining feasibility for constraints indexed by $j \in J$. In this case a problem

$$\begin{cases} \min\limits_x & g_i^+(x) \\ \text{subject to} & g_j(x) \le 0,\ j \in J \\ & x \in X \end{cases}$$

is being solved at any one time.

In all these cases, if the constraints are inconsistent, then the phase I approach terminates at a point, x' say, with an indication that the problem is infeasible. At this stage it is possible to write down an equivalent *feasibility problem* F that has been solved, whose weights depend on the type of phase I approach. The weights w_i are nonnegative and are not all zero. Infeasibility in the NLP problem is then equivalent to having obtained a solution x' of F for which the objective function is greater than zero.

$$\mathrm{F} \begin{cases} \min\limits_x & \sum\limits_{i \in J^\perp} w_i g_i^+(x) \\ \text{subject to} & g_j(x) \le 0,\ j \in J \\ & x \in X. \end{cases}$$

All of the above cases fit into this framework. (In the l_∞ case, there exist nonnegative weights at the solution such that $\sum w_i = 1$ and $w_i = 0$ if g_i does not attain the maximum value.)

In the context of the problem NLP(y^k) introduced in Section 3, the general feasibility problem has the form

$$\mathrm{F}(y^k) \begin{cases} \min\limits_x & \sum\limits_{i \in J^\perp} w_i^k g_i^+(x, y^k) \\ \text{subject to} & g_j(x, y^k) \le 0,\ j \in J \\ & x \in X \end{cases}$$

where y^k is some fixed assignment of the integer variables y of problem P and the weights w_i^k may differ for different y^k. An important property of F(y^k) is expressed in the following lemma.

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

$$\sum_{i \in J^\perp} w_i^k (g_i^k)^+ > 0 \tag{1}$$

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

$$0 \ge g_i^k + (\nabla g_i^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \qquad \forall i \in J^\perp$$

$$0 \ge g_j^k + (\nabla g_j^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \qquad \forall j \in J,$$

for all $x \in X$.

Proof:
Assume that y^k is feasible in the above constraints, so that there exists an $\hat{x} \in X$ such that $(\hat{x}, y^k)$ satisfies the following set of inequalities:

$$0 \geq g_i^k + (\nabla g_i^k)^T \begin{pmatrix} \hat{x} - x^k \\ 0 \end{pmatrix} \forall i \in J^\perp \tag{2}$$

$$0 \geq g_j^k + (\nabla g_j^k)^T \begin{pmatrix} \hat{x} - x^k \\ 0 \end{pmatrix} \forall j \in J. \tag{3}$$

It is convenient to handle the condition $\hat{x} \in X$ by introducing the normal cone $\partial X(x^k)$ at x^k. The normal cone $\partial X(x^k)$ is defined as the set of vectors u that satisfy the inequality

$$u^T(x - x^k) \leq 0 \qquad\qquad \forall x \in X. \tag{4}$$

The first order Kuhn–Tucker conditions for the feasibility problem $\mathrm{F}(y^k)$ imply that there exist multipliers $\lambda_j \geq 0,\ j \in J$ and a vector $u \in \partial X(x^k)$ such that

$$\lambda_j g_j^k = 0,\ \forall j \in J \tag{5}$$

$$\sum_{i \in J^\perp} w_i^k \nabla g_i^k + \sum_{j \in J} \lambda_j \nabla g_j^k + u = 0. \tag{6}$$

After multiplying (2) by $w_i^k \geq 0$ and (3) by $\lambda_j \geq 0$, summing, and adding to (4) the following inequality is obtained.

$$0 \geq \sum_{i \in J^\perp} w_i g_i^k + \sum_{j \in J} \lambda_j g_j^k + [\ \sum_{i \in J^\perp} w_i \nabla g_i^k + \sum_{j \in J} \lambda_j \nabla g_j^k + \begin{pmatrix} u \\ 0 \end{pmatrix}]^T \begin{pmatrix} \hat{x} - x^k \\ 0 \end{pmatrix}. \tag{7}$$

Substituting (5) and (6) the inequality (7) becomes

$$0 \geq \sum_{i \in J^\perp} w_i^k g_i^k$$

which contradicts (1) and proves the lemma.
□

We are grateful to an anonymous referee for observing that the lemma can be strengthened. The proof shows that it suffices to consider those inequalities (2) and (3) for which $w_i > 0$ and $\lambda_j > 0$ respectively. Or in other words it is sufficient to consider only the *strongly active* inequalities.

3 Reformulation of P

In this section it is shown how the model problem P can be reformulated using outer approximation to obtain an equivalent MILP master program M, relaxations of which will be used in the algorithms in Section 4. This generalizes the outer approximation algorithm of Duran and Grossmann [2] to functions that are nonlinear in the integer variables y, simplifying a similar attempt by Yuan et. al. [10]. Additionally the present approach corrects an inaccuracy which occurs in [2] and [10] when treating infeasible subproblems. Although Duran and Grossmann acknowledge the fact that their proposed algorithm may cycle if infeasible subproblems are encountered, the solution they propose (eliminate integer assignments through the use of integer cuts) is only practical for binary

variables. Moreover, it does not recognise the fact that this shortcoming is caused by a wrong interpretation of the master program. This section shows how any common Phase I approach to NLP, based on the feasibility problem of Section 2, can be used to provide a master program which is correctly equivalent to P. Another possibility is to use an exact penalty function and the benefits of this are explained in Section 5.

Earlier attempts to deal with the difficulties arising from infeasible subproblems include the work by Viswanathan and Grossmann in [9]. As far as we know, however, no proof has yet been given that these methods guarantee that no integer assignment is visited twice. It is suggested in [9] that the problem P be reformulated as

$$\left\{ \begin{array}{ll} \min\limits_{x\ y\ \alpha} & f(x,y) + \sigma\alpha \\ \text{subject to} & g(x,y) \ \leq \ \alpha \\ & \alpha \geq 0,\ x \in X,\ y \in Y \text{ integer} \end{array} \right.$$

where σ is a large positive constant. It is shown in Section 5 that this can be included as a special case of a more widely applicable formulation.

In reformulating P, the first step is to express P in terms of a projection on to the y variables, that is

$$\text{proj(P)} \left\{ \ \min_{y^j \in V} \{\text{NLP}(y^j)\}. \right.$$

where

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

is the set of all integer assignments y that give rise to feasible subproblems. In this projection the subproblem

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

is defined in which the integer variables are fixed at the value $y = y^j$. Let x^j denote a solution of NLP(y^j) for $y^j \in V$ (existence of x^j follows by the compactness of X). Because a constraint qualification (assumption **A3**) holds at the solution of every subproblem NLP(y^j) for every $y^j \in V$, it follows that proj(P) has the same solution as the problem

$$\min_{y^j \in V} \left\{ \begin{array}{ll} \min\limits_{x} & f^j + (\nabla f^j)^T \left(\begin{array}{c} x - x^j \\ 0 \end{array} \right) \\ \\ \text{subject to} & 0 \geq g^j + [\nabla g^j]^T \left(\begin{array}{c} x - x^j \\ 0 \end{array} \right) \\ & x \in X \end{array} \right\}.$$

In fact it suffices to include those linearizations of constraints about (x^j, y^j) which are *strongly active* (constraints with nonzero multipliers) at the solution of the corresponding subproblem. This is important since it implies that fewer constraints will have to be added to the master program in Section 4.

It is convenient to introduce a dummy variable $\eta \in \mathbb{R}$ into this problem, giving rise

to the equivalent problem

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

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

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

is obtained, where $T = \{j : \mathrm{NLP}(y^j)$ is feasible and x^j is an optimal solution to $\mathrm{NLP}(y^j)\}$.

It remains to find a suitable representation of the constraint $y \in V$ by means of supporting hyperplanes. The master problem given in [2] is obtained from problem M_V by replacing $y \in V$ by $y \in Y$. Duran and Grossmann justify this step by arguing that a representation of the constraints $y \in V$ is included in the linearizations in problem M_V. This argument is erroneous as the following example indicates. Consider the problem

$$\mathrm{P} \left\{ \begin{array}{ll} \min\limits_{x,y} & f(x,y) = -2y - x \\ \text{subject to} & 0 \geq x^2 + y \\ & y \in \{-1, 1\} \end{array} \right.$$

with solution $(x^*, y^*) = (1, -1)$ and $f^* = 1$. The master program given in [2] can be written as

$$\mathrm{M}_{DG} \left\{ \begin{array}{ll} \min\limits_{x,y} & f(x,y) = -2y - x \\ \text{subject to} & 0 \geq 1 + 2(x-1) + y \\ & y \in \{-1, 1\} \end{array} \right.$$

and problem M_{DG} has the solution $(x^*, y^*) = (0, 1)$ and $f^* = -2$, which is infeasible in P. Thus the value $y = 1$ is not excluded in M_{DG} and hence M_{DG} and P are not equivalent in this case.

This small example clearly illustrates that it is necessary to include information from infeasible subproblems. However care has to be taken when choosing the value of x about which to linearize the subproblem, as is illustrated by choosing $x = \frac{1}{2}$ for the infeasible subproblem $y = 1$. This choice results in the constraint

$$\frac{1}{4} + (x - \frac{1}{2}) + y \leq 0$$

being added to M_V, which does not exclude $y = 1$ from M_V.

It is necessary to ensure that integer assignments which produce infeasible subproblems are also infeasible in the master program M. Let the integer assignment y^k produce an infeasible subproblem and denote

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

It then follows directly from Lemma 1 of Section 2 that the constraints

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

exclude all integer assignments y^k for which 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 F(y^k) when infeasible subproblems are obtained, giving rise to the following MILP master problem.

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

In the above example a suitable form of feasibility problem F is

$$\min_x (x^2 + 1)^+$$

which is solved by $x = 0$. Thus the constraint

$$y \leq 0$$

is added to M_{DG} which correctly excludes the infeasible integer assignment $y = 1$.

The above development provides a proof of the following result:

Theorem 1 *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 NLP(y^j) to be solved first. This would be a very inefficient way of solving problem P. Another practical disadvantage of M is that it contains a very large number of constraints. For example if $Y = \{0, 1\}^p$ and P has m constraints then M would contain $2^p \cdot (m + 1)$ constraints. 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.

4 The Algorithms

In this section a new *linear outer approximation* algorithm is developed, based on solving MILP relaxations of the master problem M of Section 3, and it is proved that the algorithm terminates finitely. The algorithm owes much to the innovative work of Duran and

Grossmann [2] in developing an outer approximation algorithm, but the new algorithm improves on this work in a number of ways. The problem formulation allows the integer variables y to occur nonlinearly in f and g, and a new and more simple proof of termination is given. The occurrence of infeasible solutions to NLP subproblems is treated in a rigorous way which is generally applicable to many different methods for solving Phase I problems. The resulting method is also suitable for pure INLP problems in which the x variables in P are absent, which is not the case for the Duran and Grossmann formulation.

The practical performance of the resulting algorithm has proved to be similar to that of the Duran and Grossmann algorithm. However, the worst case performance of the algorithm is studied and an example is provided which shows that the algorithm can be very inefficient. This subsequently motivates us to investigate a *quadratic outer approximation* algorithms which solve MIQP master problems in an attempt to take second order information into account.

Each iteration of the linear outer approximation algorithm chooses a new integer assignment y^i and attempts to solve 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 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

$$T^i = \{j \mid j \leq i : \text{NLP}(y^j) \text{ is feasible and } x^j \text{ is an optimal solution to NLP}(y^j)\}$$
$$S^i = \left\{k \mid k \leq i : \text{NLP}(y^k) \text{ is infeasible and } x^k \text{ solves } F(y^k)\right\}.$$

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

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

where

$$\text{UBD}^i = \min f^j,\ j \leq i,\ j \in T^i.$$

Thus the following master problem is defined

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

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 is given; set $i = 0$, $T^{\perp 1} = \emptyset$, $S^{\perp 1} = \emptyset$ and UBD $= \infty$.

REPEAT

1. Solve the subproblem NLP(y^i), or the feasibility problem F(y^i) if NLP(y^i) is infeasible, and let the solution be x^i.

2. Linearize the objective and (active) constraint functions about (x^i, y^i). Set $T^i = T^{i\perp 1} \cup \{i\}$ or $S^i = S^{i\perp 1} \cup \{i\}$ as appropriate.

3. **IF** (NLP(y^i) is feasible and $f^i <$ UBD) **THEN**
update current best point by setting $x^* = x^i$, $y^* = y^i$, UBD $= f^i$.

4. Solve the current relaxation M^i of the master program M, giving a new integer assignment y^{i+1} to be tested in the algorithm. Set $i = i + 1$.

UNTIL (M^i is infeasible).

The algorithm also detects whether or not P is infeasible. If UBD$= \infty$ on exit then all integer assignments $y \in Y$ were visited by the algorithm and none was found to be feasible (i.e. the upper bound did not work as a cut off). 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 2 *If assumptions* **A1**, **A2** *and* **A3** *hold, and* $|Y| < \infty$, *then Algorithm 1 terminates in a finite number of steps at an optimal solution of P or with an indication that P is infeasible.*

Proof:
First it is shown that no integer assignment is generated twice by the algorithm. The finiteness of Algorithm 1 then follows from the finiteness of the set Y. Let $l \leq i$. If $l \in S^i$ it follows from Lemma 1 of Section 2 that the cuts introduced from the solution of the feasibility problem F(y^l) exclude y^l from any subsequent master program.
If $l \in T^i$ it is assumed that y^l is feasible in M^i and a contradiction is sought. Solving M^i gives the solution $(\eta^{i+1}, \hat{x}^{i+1}, y^l)$, which must satisfy the following set of inequalities:

$$\eta^{i+1} < \text{UBD}^i \leq f^l \tag{8}$$

$$\eta^{i+1} \geq f^l + (\nabla f^l)^T \begin{pmatrix} \hat{x}^{i+1} - x^l \\ 0 \end{pmatrix} \tag{9}$$

$$0 \geq g^l + [\nabla g^l]^T \begin{pmatrix} \hat{x}^{i+1} - x^l \\ 0 \end{pmatrix}. \tag{10}$$

Since x^l is the optimal solution to NLP(y^l) and a constraint qualification holds (**A3**), no feasible descent direction exists at x^l, that is

$$0 \geq g^l + [\nabla g^l]^T \begin{pmatrix} \hat{x}^{i+1} - x^l \\ 0 \end{pmatrix}$$

$$\Rightarrow (\nabla f^l)^T \begin{pmatrix} \hat{x}^{i+1} - x^l \\ 0 \end{pmatrix} \geq 0. \tag{11}$$

Substituting (11) into (9) gives

$$\eta^{i+1} \geq f^l$$

which contradicts (8). Thus y^l is infeasible for all $l \in S^i$ and $l \in T^i$.

Finally it is shown that Algorithm 1 always terminates at a solution of P or with an indication that P is infeasible. If P is feasible, then let an optimal solution to P be given by (x^*, y^*) with optimal value f^* (any other optimal solution has the same objective value and the algorithm does not distinguish between them). Since M is a relaxation of P, f^* is an upper bound on the optimal value of M, which is attained at (x^*, y^*). Now assume that the algorithm terminates with an indicated solution (x', y') with $f' > f^*$ (i.e. not optimal). Since UBD$= f' > f^*$ it follows that (x^*, y^*) must be feasible in the previous relaxation of M, which contradicts the assumption that the algorithm terminates at (x', y'). If on the other hand P is infeasible then all NLP(y^j)–subproblem are infeasible and the algorithm never updates the upper bound UBD, and hence exits with UBD$= \infty$ indicating an infeasible problem.
□

It can be observed from the proof that it is not necessary to solve M^i for optimality in Algorithm 1, as long as a new integer assignment is obtained from M^i. However, if M^i is solved for optimality then the upper bound on η can be supplemented by a weak lower bound

$$\eta \geq \eta^i$$

where η^i is the solution value of $M^{i\perp 1}$. This lower bound can improve the efficiency of the MILP solver by cutting out branches of the branch and bound tree that need not be examined. It is worth remarking that the method of proof used here is much more simple (especially with respect to the derivation of the master problem M) than that of Duran and Grossmann which is based on integer polyhedra and linear programming theory.

There are a number of practical considerations that arise when implementing the algorithm. It is mentioned in Section 3 that it is worthwhile to include only those constraints that are active at a solution of the subproblem NLP(y^i) so that fewer linearizations are added to the master program at each iteration. If this is done it might not be necessary to include a constraint dropping procedure that scans the constraints of the master program to keep its size small. On the other hand, adding fewer constraints to the master program implies that the master program relaxations are weaker which could result in a larger number of iterations. Currently we prefer to add only the linearizations of the strongly active constraints to keep the size of the master program smaller. Preliminary computational experience shows that this results in better overall CPU times whilst not incurring many additional outer approximation iterations.

In practice the constraint

$$\eta < \text{UBD}$$

would not be used, but rather

$$\eta \leq \text{UBD} - \epsilon$$

where ϵ is some small user supplied accuracy. The algorithm can then only be guaranteed to provide an ϵ–optimal solution to P. As mentioned in Section 3 our reformulation includes pure INLP problems and this makes Algorithm 1 applicable to pure INLP problems, in which case step 1 of the algorithm (the inner optimization over the continuous variables) becomes redundant.

Practical experience with linear outer approximation given in [2] indicates that outer approximation is superior both to nonlinear branch and bound and Generalized Benders Decomposition, although the test problems are limited to the case where f and g are

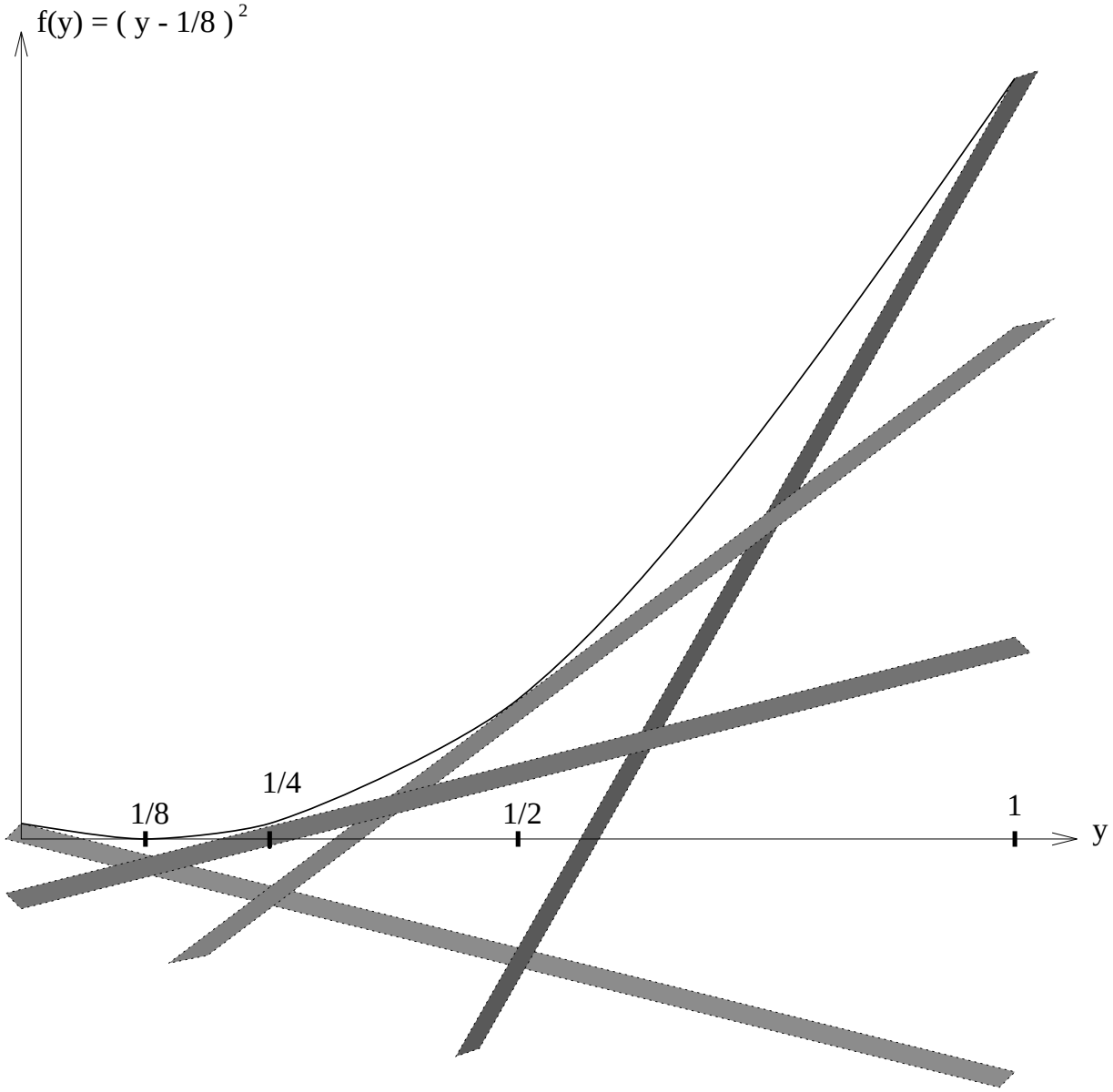

Figure 1: Worst case example for Algorithm 1

both linear functions in y. It is of interest to know whether this is always the case, or if there exist situations in which the outer approximation algorithm performs badly. We have been able to construct a *worst case example* for linear outer approximation in which Algorithm 1 visits all integer feasible points in the problem before finding the solution, even though the initial assignment y^0 is adjacent to the optimal assignment. The example is

$$\begin{cases} \min\limits_{y} & f(y) = (y-\epsilon)^2 \\ \text{subject to} & y \in \{0, \epsilon, \ldots, \frac{1}{2}, 1\} \end{cases}$$

in which $\epsilon = 2^{\perp p}$ for some $p > 1$.

Starting with $y^0 = 0$, which is the adjacent value to the solution $y^* = \epsilon$, the next iterate is $y^1 = 1$, which is an extreme feasible point. Algorithm 1 then works its way back to the solution by visiting each remaining integer assignment $y^i = 2^{\perp i+1}$, $i = 2, 3, \ldots, p+1$ in turn. Figure 1 illustrates the situation for $p = 3$ and the shaded boxes indicate the various supporting hyperplanes that are generated. This example is also a worst case

example for Generalized Benders Decomposition but it is solved by nonlinear branch and bound in only one step. The problem could also be slightly perturbed to $f(y) = (y - \delta)^2$ with $\delta < \frac{\epsilon}{2}$. Then starting at the solution $y^0 = 0$, linear outer approximation would again visit all feasible points before verifying that y^0 is the solution. Again Generalized Benders Decomposition also visits all feasible points, but nonlinear branch and bound solves the problem after one branch.

The example shows that both linear outer approximation and Generalized Benders Decomposition perform badly when the problem functions are not adequately represented by linear approximations. The initial step makes the next iterate remote from the solution which is unsatisfactory in a nonlinear situation. The remedy lies in introducing 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. The resulting algorithm is referred to as *quadratic outer approximation* and is obtained by replacing the relaxed master problem M^i by the problem

$$(Q^i) \begin{cases} \min\limits_{x,y,\eta} & \eta + \frac{1}{2} \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix}^T [\nabla^2 \mathcal{L}^i] \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix} & \\ \text{subject to} & \eta < UBD & \\ & \eta \geq f^j + (\nabla f^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} & \forall j \in T^i \\ & 0 \geq g^j + [\nabla g^j]^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} & \\ & 0 \geq g^k + [\nabla g^k]^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} & \forall k \in S^i \\ & x \in X,\ y \in Y \text{ integer} & \end{cases}$$

in step 4 of Algorithm 1. In the definition of Q^i the function

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

is the usual Lagrangian function.

Including a curvature term in the objective function does not change the finite convergence property expressed in Theorem 1, since the feasible region of M^i is unchanged and the constraints of Q^i are still supporting hyperplanes. However, the possibility of using the lower bound $\eta \geq \eta^i$ is no longer conveniently available. A quadratic Taylor series does not provide a lower bound on a convex function as the linear Taylor series does. Therefore it is not possible to use the optimal value of Q^i as a lower bound. Moreover it cannot even be expected that the optimal value of the linear part of the objective function (η) of the master problem relaxation provides a lower bound on P although η is only constrained by the supporting hyperplanes on f. This is illustrated by the following example:

$$\begin{cases} \min\limits_{x} & f(x) = -\ln(x + 1) \\ \text{subject to} & 0 \leq x \leq 2. \end{cases}$$

For $x = 0$ the minimum of the quadratic approximation to f is at $x = \frac{1}{2}$ and the value of the linear approximation about $x = 0$ at $x = \frac{1}{2}$ is $-\frac{1}{2}$ which is greater than the minimum of the above problem ($-\ln(3)$).

The advantage of the quadratic outer approximation algorithm is that a possibly different selection y^{i+1} is made by solving the master problem Q^i, which takes into account

nonlinear terms in P. This is well seen in the worst case example for linear outer approximation, which is solved by the quadratic outer approximation algorithm in two iterations, independent of p. The price that has to be paid for this better performance is that instead of solving an MILP at each iteration, an MIQP master program has to be solved. Unfortunately there is little or no software available that is specifically tailored to an MIQP problem. The authors are currently investigating ways to solve the resulting MIQP problems by a branch and bound strategy that uses improved lower bounds for problems generated by branching. A Generalized Benders Decomposition approach for solving the MIQP as suggested by Lazimy [8] seems inadequate, since Generalized Benders Decomposition can again be interpreted as a linear model, in which case the difficulties caused by nonlinearities in outer approximation will simply arise at the MIQP level. Other methods that have been suggested include the branching rule of Körner [7] and a branching rule that was suggested by Breu and Burdet [1] for linear 0–1 programming.

In order to gain further insight into the proposed algorithms it is useful to consider the case when they are applied to pure integer nonlinear problems. Both algorithms make linear approximations to the feasible constraints at y^i; the difference lies in the fact that the quadratic algorithm also includes a second order Lagrangian term in the next master program. Therefore quadratic outer approximation can be interpreted as a Sequential Quadratic Programming method generalized to integer programming. Linearizations of previous steps are kept in the master program to avoid cycling between successive integer assignments, and the QP subproblem of an ordinary SQP method is replaced by an MIQP problem to account for the discrete nature of the problem. The linear outer approximation algorithm can be interpreted as a Sequential Linear Programming technique.

This observation gives an indication as to when quadratic outer approximation should be preferred. If y appear only linearly in the problem, then it is hardly worthwhile to solve MIQP master programs. There is also unlikely to be much advantage in using quadratic outer approximation when the integer variables are mostly zero–one variables. The most favourable case for quadratic outer approximation occurs when there are multiple discrete values of each component of y, and there are nonlinear terms in y present. However, care has to be taken, since nonlinearities can be hidden away by using linearization techniques that reformulate the original problem. Such a reformulation would indeed be necessary if one wanted to use Duran and Grossmann's outer approximation algorithm. It is hoped to present results with the quadratic outer approximation algorithm in a later paper in conjunction with an improved branch–and–bound procedure for MIQP problems.

5 Nonsmooth MINLP problems

In this section the OA algorithms of Section 4 are generalized to cover an MINLP problem applicable to nonsmooth functions. Similarly to Section 3, this problem can be reformulated using projection and first order conditions to obtain a (nonsmooth) master problem. This problem is equivalent to an MILP problem if extra variables are added. Relaxations of this master program are used in deriving an OA algorithm which iterates finitely between nonsmooth NLP subproblems and MILP master program relaxations. Exact Penalty functions form a subclass of the type of optimization problems considered here and their use in OA is further examined. The main attraction of exact penalty functions lies in the fact that they make a distinction between feasible and infeasible subproblems unnecessary. Sufficient conditions are given under which the MINLP and its exact penalty function formulation are equivalent. Particular attention is given to the l_1 exact penalty problem and it is shown in this case how extra variables can be used to convert the nonsmooth master problem to an MILP problem. An alternative version of

outer–approximation is developed, where only one cut is added to the master program relaxations per iteration and it is shown that this cut can be equivalent to the corresponding Benders cut. Although only an MILP version of the algorithm is developed here, it is noted that the same observations as in Section 4 with respect to its performance are valid here and it is indicated how a curvature term can be included in the MILP master program relaxations.

The class of problem considered here is

$$\mathrm{P}_{ns} \left\{ \begin{array}{ll} \min\limits_{x,y} & f(x,y) + h(g(x,y)) \\ \text{subject to} & x \in X \ , \ y \in Y \text{ integer} \end{array} \right.$$

where f and g are continuously differentiable and X is as in **A1**. It is assumed that $h : \mathbb{R}^m \to \mathbb{R}$ is convex but nonsmooth. This assumption alone does not imply convexity of P_{ns} which is needed to enable its treatment by outer approximation. It is therefore convenient to assume that h is also a monotone function, that is

$$a \leq b \ \Rightarrow \ h(a) \leq h(b).$$

This class of problems includes a wide range of practical optimization problems such as problems involving exact penalty functions which are studied in more detail towards the end of this section. In many cases $h(g)$ is a polyhedral convex function such as $h(g) = \max_i g_i$, $h(g) = \|g^+\|_\infty$ or $h(g) = \|g^+\|_1$, but other functions are also possible. (Here a^+ denotes the vector $a^+ = (a_1^+, \ldots, a_m^+)^T$ where $a_i^+ = \max(a_i, 0)$.)

To reformulate P_{ns} similar techniques to those used in Section 3 are employed. First a projection onto the integer variables

$$\mathrm{proj}(\mathrm{P}_{ns}) \left\{ \begin{array}{l} \min\limits_{y^j \in Y} \{\mathrm{NSO}(y^j)\}. \end{array} \right.$$

is defined, where the nonsmooth subproblem $\mathrm{NSO}(y^j)$ is obtained from P_{ns} by fixing the integer variables at $y = y^j$, that is

$$\mathrm{NSO}(y^j) \left\{ \begin{array}{ll} \min\limits_{x} & f(x,y^j) + h(g(x,y^j)) \\ \text{subject to} & x \in X. \end{array} \right.$$

Let x^j be an optimal solution of $\mathrm{NSO}(y^j)$. As a consequence of the subgradient inequality and the first order necessary conditions ([3], Theorem 14.6.1, p. 406 f.) $\mathrm{NSO}(y^j)$ has the same solution as the following linearized problem.

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

where a dummy variable η has been introduced. Let $\eta^j = f^j + h(g^j)$ denote the optimal value of η. Replacing $\mathrm{NSO}(y^j)$ by its linearization implies that the projected problem $\mathrm{proj}(\mathrm{P}_{ns})$ has the same solution as

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

Next we define the master program

$$\mathrm{M}_{ns}\left\{\begin{array}{lll}\min\limits_{\eta,x,y} & \eta & \\ \text{subject to} & \eta \geq f^j + (\nabla f^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} + h(g^j + [\nabla g^j]^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix}) & j \in T \\ & x \in X\ ,\ y \in Y \text{ integer} & \end{array}\right.$$

where $T = \{j : x^j \text{ is an optimal solution to NSO}(y^j)\}$. It readily follows that (η^i, x^i, y^i) is feasible in M_{ns} for any $i \in T$. (Proof: use $\eta^i \geq f^i + h(g^i)$ and expand about (x^i, y^i) using convexity and monotonicity of h.) The above development shows that M_{ns} is equivalent to P_{ns} in the sense that (x^*, y^*) solves P_{ns} if and only if (η^*, x^*, y^*) solves M_{ns}.

The master program M_{ns} is not solved directly but instead a relaxation strategy similar to Algorithm 1 is applied. The relaxation M^i_{ns} that is solved at iteration i of the algorithm is obtained as

$$M^i_{ns}\left\{\begin{array}{lll}\min\limits_{\eta,x,y} & \eta & \\ \text{subject to} & \eta < \mathrm{UBD} & \\ & \eta \geq f^j + (\nabla f^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} + h(g^j + [\nabla g^j]^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix}) & j \in T^i \\ & x \in X\ ,\ y \in Y \text{ integer.} & \end{array}\right.$$

where $T^i = \{j \leq i : x^j \text{ is an optimal solution to NSO}(y^j)\} \subset T$ and

$$\mathrm{UBD} = \min_{j \leq i}\{f^j + h(g^j)\}.$$

The program M^i_{ns} can now be used in an outer approximation algorithm similar to Algorithm 1. The only unusual feature is the occurrence of the convex composition $h(g)$. However, using standard linear programming techniques, $h(g)$ can be expressed as a set of linear inequality constraints if h is a polyhedral function, and this is described below for the l_1 norm. The new algorithm can now be described as follows.

Algorithm 2: Nonsmooth Outer Approximation

Initialization: y^0 is given; set $i = 0$, $T^{\perp 1} = \emptyset$ and $\mathrm{UBD} = \infty$.

REPEAT

1. Solve the subproblem $\mathrm{NSO}(y^i)$ and let the solution be x^i.
2. Linearize the objective and (active) constraint functions about (x^i, y^i).
   Set $T^i = T^{i\perp 1} \cup \{i\}$.
3. **IF** $(f^i + h(g^i) < \mathrm{UBD})$ **THEN**
   update current best point by setting $x^* = x^i$, $y^* = y^i$, $\mathrm{UBD} = f^i + h(g^i)$.
4. Solve the current relaxation M^i_{ns} of the master program M_{ns}, giving a new integer assignment y^{i+1} to be tested in the algorithm. Set $i = i + 1$.

UNTIL (M^i_{ns} is infeasible).

The following theorem establishes the finite convergence of the algorithm.

Theorem 3 *If assumptions* **A1** *and* **A2** *are satisfied and Y is finite then Algorithm 2 converges finitely to a solution of P_{ns}.*

Proof:
It is shown first that no integer assignment is generated twice by the algorithm. Its finiteness then follows from the finiteness of Y.

It is assumed that at iteration $i \geq j$ the integer assignment y^j is feasible in the master program M^i_{ns} and a contradiction is sought. It follows that there exists an $x' \in X$ satisfying the inequality

$$\eta \geq f^j + (\nabla f^j)^T \begin{pmatrix} x' - x^j \\ 0 \end{pmatrix} + h(g^j + [\nabla g^j]^T \begin{pmatrix} x' - x^j \\ 0 \end{pmatrix}).$$

Let $\lambda^j \in \partial h(g^j)$ be the optimal multiplier of NSO(y^j). It follows from the definition of the subdifferential ∂h that

$$\eta \geq f^j + h(g^j) + (\nabla f^j + \nabla g^j \lambda^j)^T \begin{pmatrix} x' - x^j \\ 0 \end{pmatrix}. \tag{12}$$

(Because h is monotonic it follows that $h(g(x,y))$ is convex and $\nabla g^j \lambda^j$ is an element of its subdifferential.)

In order to apply the optimality conditions of Theorem 14.6.1 ([3] p. 406 f.) it is convenient to handle the constraint $x' \in X$ by introducing composite functions. Since X contains only linear functions like $r_i(x) = r_i^T x - b_i \leq 0,\ i = 1, \ldots, q$ these constraints can be fitted into the framework of the above optimality conditions through the single constraint

$$t(r(x)) \leq 0$$

involving the polyhedral function

$$t(r(x)) = \max_i r_i(x).$$

The optimality of x^j implies the existence of multipliers $\pi^j \geq 0$ and $\mu^j \in \partial t(r(x^j))$. Premultiplying the linear constraint by $\pi^j \mu^j_i$, summing over all $i = 1, \ldots, q$ and adding to (12) gives the following valid inequality

$$\eta \geq f^j + h(g^j) + t(r^j)\pi^j + (\nabla f^j + \nabla g^j \lambda^j + \nabla r^j \mu^j)^T \begin{pmatrix} x' - x^j \\ 0 \end{pmatrix}.$$

The first order necessary conditions for NSO(y^j) (e.g. [3], Theorem 14.6.1, p. 406f.) imply that

$t(r^j)\pi^j = 0$	*complementarity*
$\nabla f^j + \nabla g^j \lambda^j + \nabla r^j \mu^j = 0$	*1st order condition.*

Thus the inequality

$$\eta \geq f^j + h(g^j)$$

can be derived for η. This contradicts the strict upper bound on η which is

$$\eta < \text{ UBD } \leq f^j + h(g^j).$$

Now assume that Algorithm 2 terminates with an indicated solution for which

$$\text{UBD } = f' + h(g') > f^* + h(g^*).$$

The convexity assumption implies that y^* must be feasible in the previous MILP master program relaxation which contradicts the termination assumption and concludes the proof.
□

It is worth mentioning, that Algorithm 2 does not require a constraint qualification on g to hold in order to achieve finite convergence. However, such an assumption is needed to show that the exact penalty function formulation of P and P itself are equivalent and we now proceed to examine this situation.

A class of nonsmooth MINLP which is of particular interest are exact penalty functions (EPFs). EPFs offer an alternative approach to the difficulties caused by infeasible subproblems which makes a distinction between feasible and infeasible NLP subproblems unnecessary. Instead of solving problem P an exact penalty function formulation of P is considered.

$$\mathrm{E}\left\{\begin{array}{ll}\min\limits_{x,y} & \Phi(x,y) = \nu f(x,y) + \|g(x,y)^+\| \\ \text{subject to} & x \in X,\ y \in Y \text{ integer.}\end{array}\right.$$

where $\|\cdot\|$ is a norm in $\mathbb{R}^m$, and ν is a sufficiently small penalty parameter. This is a special case of P_{ns} and can be solved by Algorithm 2.

It is of interest to know under which conditions E and P are equivalent. Theorem 4 gives sufficient conditions under which the mixed integer exact penalty function problem E is equivalent to problem P, so that Algorithm 2 terminates at a solution to P. One of these conditions is that the penalty parameter has to be "sufficiently small". This is qualified by the following conditions on the penalty parameter, where $\|\cdot\|_D$ denotes the dual norm to $\|\cdot\|$.

A4 Let the penalty parameter ν satisfy

$$\nu < 1/\max_j \|\lambda^j\|_D \qquad \forall j:\ \mathrm{NLP}(y^j) \text{ is feasible}$$

$$\nu < \frac{\|(g^k)^+\|}{f^* \perp f^k} \qquad \forall k: \|(g^k)^+\| > 0 \text{ and } f^k < f^*.$$

A5 Let a second order sufficient condition (e.g. [3], Theorem 9.3.2, p. 211) hold for all j such that $\mathrm{NLP}(y^j)$ is feasible.

Although additional assumptions have to be made **A5** will usually hold. If the users choice of the penalty parameter does not satisfy **A4** then the optimal solution of E is not feasible in P. The user can detect this fact and reduce the penalty parameter accordingly.

The first condition in **A4**, together with **A5** is needed to ensure that the solution of the feasible NLP–subproblems and the corresponding EPF–problems are equivalent, and the second condition in **A4** ensures that Algorithm 2 does not terminate with an infeasible solution. A simple conclusion of Theorem 4 is that Algorithm 2 terminates finitely at a solution of P or, if P is infeasible, it finds the "best" exact penalty solution to P. Now Theorem 4 can be stated

Theorem 4 *If assumptions* **A1** *to* **A5** *hold and if P has a feasible solution, then E and P are equivalent in the sense that* (x^*, y^*) *solves P if and only if it solves E.*

Proof:
Assumptions **A3**, **A5** and the first part of assumption **A4** imply that any feasible $\mathrm{NLP}(y^j)$ subproblem of P is equivalent to the corresponding $\mathrm{NSO}(y^j)$ subproblem of E (c.f. [3], Theorem 14.3.1, p. 380). It remains, therefore, to show that the solution of E cannot be a point (x^k, y^k) for which $\mathrm{NLP}(y^k)$ is infeasible. Now let (x^k, y^k) be such that $\|(g^k)^+\| > 0$. The second part of assumption **A4** implies that

$$\begin{aligned}\Phi^k &= \nu f^k + \|(g^k)^+\| \\ &> \nu f^k + \nu(f^* - f^k) \\ &= \nu f^*\end{aligned}$$

Therefore, $\Phi^k > \Phi^*$ which concludes the proof.
□

If $h(g)$ is a polyhedral convex function, it is possible to reformulate the constraints in M_{ns} using a standard linear programming technique. If $h(g) = \|g^+\|_1$, then additional variables ξ_l are introduced and the constraints are equivalent to

$$\left.\begin{array}{lll} \eta & \geq f^j + (\nabla f^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} + \sum_{l=1}^{m} \xi_l & \\ \xi_l & \geq g_l^j + (\nabla g_l^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} & l = 1, \ldots, m \\ \xi_l & \geq 0 & l = 1, \ldots, m \end{array}\right\} (C_1)$$

In the case of $h(g) = \|g^+\|_\infty$ only a single additional variable is needed.

An alternative way of deriving the constraints C_1 is now explained in the context of the l_1 EPF problem. It is possible to introduce variables ξ_l directly into E so that it can be reformulated as

$$\mathrm{E}_1 \left\{ \begin{array}{lll} \min\limits_{x,y,\sigma} & \nu f(x,y) + \sum_{l=1}^{m} \xi_l & \\ \text{subject to} & \xi_l \geq g_l(x,y) & l = 1, \ldots, m \\ & \xi_l \geq 0, \ \forall l, \ x \in X, \ y \in Y \text{ integer}, & \end{array}\right.$$

Outer approximations of E_1 can be derived using the methods of Section 4, giving rise to the constraints C_1 (with f replaced by νf). A similar formulation is again possible for the l_∞ norm.

The proof of Theorem 3 indicates that it is possible to derive a version of Algorithm 2 in which only one constraint is added per iteration. This single cut is given by

$$\eta \geq f^j + h(g^j) + (\nabla f^j + [\nabla g^j]\lambda^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix}$$

where $\lambda^j \in \partial h(g^j)$ is the optimal multiplier vector of the NSO(y^j) subproblem. It is instructive to compare this cut to the Benders cut for the same problem. Flippo et. al. [4] show that the Benders cut can be written as

$$\eta \geq f^j + h(g^j) + \mu^j (y^j - y)$$

where μ^j is the optimal multiplier of the constraint $y = y^j$ in

$$\mathrm{P}_{ns}(y^j) \left\{ \begin{array}{ll} \min\limits_{x,y} & f(x,y) + h(g(x,y)) \\ \text{subject to} & y = y^j \\ & x \in X \ , \ y \in Y \end{array}\right.$$

The first order necessary conditions ([3], Theorem 14.6.1, p. 406 f.) enable an expression of μ^j in terms of ∇f^j, ∇g^j, and $\lambda^j \in \partial h(g^j)$ to be given

$$\nabla_y f^j + [\nabla_y g^j]\lambda^j + \mu^j = 0$$

so that the Benders cut can finally be written as

$$\eta \geq f^j + h(g^j) + (\nabla_y f^j + [\nabla_y g^j]\lambda^j)^T (y - y^j).$$

Clearly, if $x^j \in X$ lies in the strict interior of X or if all corresponding multipliers are zero, then also $\nabla_x f^j + [\nabla_x g^j]\lambda^j = 0$ and both cuts are equivalent. This last statement indicates that it might not be advisable to use just the single cut, since GBD is usually inferior to OA.

As explained in Section 4, outer–approximation forms a linear model of the problem P and does therefore not represent curvature information adequately. This motivates the introduction of a second order Lagrangian term into the objective function of the relaxed master programs. A similar approach is suggested here and the corresponding curvature term is

$$\nabla^2 \mathcal{L}^i = \nabla^2 f^i + \sum_{l=1}^{m} \lambda_l^i \nabla^2 g_l^i.$$

Finally it is possible to generalize problem P_{ns} even further by including a composite constraint of the form

$$t(r(x,y)) \leq 0.$$

It is possible to derive an equivalent MILP master program using similar techniques to those employed in this section. The inclusion of this additional constraint has, however, the disadvantage that it makes a separate treatment of infeasible subproblems necessary, whereas the main reason for introducing penalty functions is that this is avoided.

Acknowledgements

We would like to acknowledge the advice of the two referees whose comments enabled a much improved version of the paper to be prepared.

References

[1] R. Breu and C.-A. Burdet, "Branch–and–bound experiments in zero–one programming", *Mathematical Programming Study* **2** (1974) 1–50.

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

[3] R. Fletcher, *Practical Methods of Optimization* (John Wiley, Chichester, 1987).

[4] O.E. Flippo et. al., "Duality and decomposition in general mathematical programming", Econometric Institute, Report 8747/B, University of Rotterdam (1987).

[5] A.M. Geoffrion, "Generalized Benders Decomposition", *Journal of Optimization Theory and Applications* **10** (1972) 237-262.

[6] G.R. Kocis and I.E. Grossmann, "Global Optimization of Nonconvex MINLP in Process Synthesis", *Industrial & Engineering Chemistry Research* **27** (1988) 1407-1421.

[7] F. Körner, "A new branching rule for the branch–and–bound algorithm for solving nonlinear integer programming problems", *BIT* **28** (1988) 701-708.

[8] R. Lazimy, "Improved Algorithm for Mixed-Integer Quadratic Programs and a Computational Study", *Mathematical Programming* **32** (1985) 100-113.

[9] J. Viswanathan and I.E. Grossmann, "A combined penalty function and outer–approximation method for MINLP optimization", *Computers and chemical Engineering* **14** (1990) 769-782.

[10] X. Yuan, S. Zhang L. Pibouleau and S. Domenech, "Une méthode d'optimization non linéaire en variables mixtes pour la conception de procédés", *Operations Research* **22/4** (1988) 331-346.