Global Optimization of MINLP Problems in Process Synthesis and Design

C.S. Adjiman, I.P. Androulakis and C.A. Floudas[1]

Department of Chemical Engineering, Princeton University, Princeton, NJ 08544, USA

Abstract : Two new methodologies for the *global* optimization of MINLP models, the Special structure Mixed Integer Nonlinear αBB, **SMIN–αBB**, and the General structure Mixed Integer Nonlinear αBB, **GMIN–αBB**, are presented. Their theoretical foundations provide guarantees that the global optimum solution of MINLPs involving twice–differentiable nonconvex functions in the continuous variables can be identified. The conditions imposed on the functionality of the *binary* variables differ for each method : linear and mixed bilinear terms can be treated with the SMIN–αBB; mixed nonlinear terms whose continuous relaxation is twice–differentiable are handled by the GMIN–αBB. While both algorithms use the concept of a *branch & bound tree*, they rely on fundamentally different bounding and branching strategies. In the GMIN–αBB algorithm, lower (upper) bounds at each node result from the solution of convex (nonconvex) MINLPs derived from the original problem. The construction of *convex lower bounding MINLPs*, using the techniques recently developed for the generation of valid convex underestimators for twice–differentiable functions (Adjiman et al., 1996; Adjiman and Floudas, 1996), is an essential task as it allows to solve the underestimating problems to global optimality using the GBD algorithm or the OA algorithm, provided that the binary variables participate separably and linearly. Moreover, the inherent structure of the MINLP problem can be fully exploited as branching is performed on the binary and the continuous variables. In the case of the SMIN–αBB algorithm, the lower and upper bounds are obtained by solving continuous relaxations of the original MINLP. Using the αBB algorithm, these nonconvex NLPs are solved as global optimization problems and hence valid lower bounds are generated. Since branching is performed exclusively on the binary variables, the maximum size of the branch–and–bound tree is smaller than that for the SMIN–αBB. The two proposed approaches are used to generate computational results on various nonconvex MINLP problems that arise in the areas of Process Synthesis and Design.

INTRODUCTION

A wide range of chemical engineering problems can effectively be framed as *Mixed–Integer Nonlinear Problems* (MINLP) as this approach allows the simultaneous optimization of the continuous variables pertaining to a certain structure, and of the structure itself which is modeled via binary variables (Floudas, 1995; Grossmann, 1990, 1996). Such a mathematical framework has been proposed for a variety of process synthesis problems (e.g., heat recovery networks, separation systems, reactor networks), process operations problems (e.g., scheduling and design of batch processes), molecular design problems and synthesis of metabolic pathways. A number of these applications are described in Floudas (1995) and Grossmann (1996). The degree of nonconvexity of the participating functions is generally arbitrary and nonlinearities can be identified in the continuous, the integer, or joint domains. The difficulties in solving these MINLPs therefore stem not only from the combinatorial characteristics of the problem which are a direct result of the presence of the integer variables, but also from the presence of nonconvexities (Floudas and Grossmann, 1995).

A number of techniques have been developed to solve to global optimality certain classes MINLPs : the Outer Approximation algorithm, OA, and its variants (Duran and Grossmann, 1986; Kocis and Grossmann, 1987; Viswanathan and Grossmann, 1990) handle separable problems in which the binary variables participate linearly and the continuous variables participate in a convex manner; the Extended Cutting Plane method, ECP (Westerlund and Pettersson, 1995); the Generalized Outer Approximation algorithm, GOA (Fletcher and Leyffer, 1994), is applicable to problems with convex functions in the continuous and not necessarily separable binary variables; the Generalized Benders Decomposition algorithm, GBD (Geoffrion, 1972; Floudas et al., 1989), is designed for problems with a convex continuous part and binary variables in linear or mixed bilinear terms; the Generalized Cross Decomposition algorithm, GCD (Holmberg, 1990), applies to the same class as the GBD. A detailed theoretical and algorithmic description can be found in Floudas (1995). Ryoo and Sahinidis (1995) proposed reduction tests coupled with a standard branch–and–bound algorithm.

The aim of this paper is to present two new global optimization approaches for the broad class of problems represented by MINLPs involving functions that are twice–differentiable in the continuous variables. Both techniques are based on the recent development of the αBB global optimization algorithm for the treatment of twice–differentiable NLPs (Androulakis et al., 1995; Adjiman et al., 1996) , and new methods for the construction of valid convex underestimators for problems of that type

[1] Author to whom all correspondence should be addressed

(Adjiman and Floudas, 1996; Adjiman et al., 1996a,b). The SMIN–αBB algorithm is presented in the first part of this paper. Similar in essence to the αBB algorithm, it converges to the global optimum solution thanks to the ability to generate consistently tighter convex MINLPs as valid lower bounding problems. In the second part, the GMIN–αBB is discussed. It operates within an integer branch–and–bound framework, where the continuous relaxations can be solved to global optimality if needed, using the αBB algorithm for continuous NLPs. Both approaches are tested on a few small examples. In addition, the SMIN–αBB is applied to the optimization of a heat exchanger network and the GMIN–αBB is used to solve an optimal pump configuration problem.

THE SMIN–αBB ALGORITHM

The SMIN–αBB algorithm is a global optimization algorithm for MINLPs of the form :

$$\begin{array}{rll} \min\limits_{\mathbf{x}} & f(\mathbf{x}) + \mathbf{x}^T A_0 \mathbf{y} + c_0^T \mathbf{y} & \\ s.t. & \mathbf{g}(\mathbf{x}) + \mathbf{x}^T A_1 \mathbf{y} + c_1^T \mathbf{y} & \leq 0 \\ & \mathbf{h}(\mathbf{x}) + \mathbf{x}^T A_2 \mathbf{y} + c_2^T \mathbf{y} & = 0 \\ & \mathbf{x} \in X \subseteq \Re^n & \\ & \mathbf{y} \in Y = \{0,1\}^m & \end{array} \qquad (1)$$

where f, $\mathbf{g}$ and $\mathbf{h}$ belong to $\mathcal{C}^2$, the set of functions with continuous second–order derivatives, $\mathbf{x}$ is a vector of size n, $\mathbf{y}$ is a vector of size m, A_0, A_1, A_2 are $n \times m$ real matrices and c_0, c_1, c_2 are real vectors of size m.

As can be seen from (1), the binary variables can participate linearly or in bilinear mixed integer terms. Although this condition may appear restrictive at first, many other types of integer or mixed–integer terms can be transformed into this form through the introduction of additional variables.

The global optimum of a problem of type (1) is identified using a branch–and–bound scheme which allows the generation of converging sequences of valid upper and lower bounds. One of the specificities of the SMIN–αBB algorithm is that the branch–and–bound tree is constructed by branching on a combination of the continuous and binary variables. For each region of the solution space thus obtained, a convex lower bounding MINLP is derived and solved to global optimality using the OA algorithm, the GBD algorithm or the linear underestimators of Glover (1975), depending on the type of participation of the binary variables. If this problem is infeasible, or if its solution is greater than the current upper bound for problem (1), the region is fathomed. Otherwise, an upper bound is generated through the solution of the original nonconvex MINLP restricted to the current domain. The results are then used to guide further exploration of the solution space : the node with the smallest lower bound is split into two new domains. Combined with an underestimating strategy which provides gradually tighter convex lower bounding problems, this approach results in the identification of the global optimum solution with ϵ–convergence.

Underestimating Strategy

Functions participating in problems of type (1) can be separated into a purely continuous part and a mixed–integer part. Binary variable participation of the type found in the class of problems studied can be readily handled by the OA/ER/AP, in the case of linear binary terms, or the GBD in the case of mixed bilinear terms. However, if the continuous part of the functions is nonconvex, these algorithms may get trapped at a local solution. In order to get a valid lower bound on the nonconvex MINLP in a given region, the problem must therefore be *convexified* and *underestimated*. For this purpose, it suffices to convexify and underestimate the continuous part.

The automatic generation of valid convex underestimators for a general twice–differentiable function $f(\mathbf{x})$ has been studied in detail in the context of the αBB algorithm for nonconvex NLPs (Androulakis et al., 1995; Adjiman et al., 1996). Maranas and Floudas (1993) have shown that an appropriate lower bounding function $\mathcal{L}(\mathbf{x})$ for $f(\mathbf{x})$ over the region $[\mathbf{x}^L, \mathbf{x}^U]$ is given by

$$\mathcal{L}(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i (x_i^L - x_i)(x_i^U - x_i) \qquad (2)$$

where the α_i s are sufficiently large scalars. In particular, they showed that if a single α value is used, then the following condition guarantees the convexity of the underestimator :

$$\alpha \geq \max\{0, -\frac{1}{2} \min_{k, \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U} \lambda_k(\mathbf{x})\}$$

where the λ_k's are the eigenvalues of the Hessian matrix of $f(\mathbf{x})$.

A number of methods have been proposed in Adjiman et al. (1996a,b) for the calculation of a valid bound on the minimum eigenvalue of the Hessian matrix over the domain of interest. Other means to ensure the positive semi–definiteness of the Hessian matrix of the underestimator $\mathcal{L}(\mathbf{x})$ are also presented. The use of interval arithmetic in order to obtain the interval Hessian matrix of $\mathcal{L}(\mathbf{x})$ or its interval characteristic polynomial is common feature of all the procedures developed. Algorithms of varying degrees of accuracy and computational complexity are used to rigorously determine values of the α_i parameters that satisfy criteria that guarantee convexity of $\mathcal{L}(\mathbf{x})$ for a given set of variables bounds. The application of these approaches to numerous examples has shown that the constructed underestimators lead to the identification of the global optimum solution of difficult nonconvex NLPs within reasonable CPU times. Using these techniques, a convex underestimating MINLP can therefore be derived for any problem of type (1). In addition, an increase in the quality of the underestimators as variable bounds narrow is inherent in all of these approaches so that a *nondecreasing* sequence of lower bounds is indeed obtained.

Branching Strategies

While the underestimating strategy is essential to ensure the identification of the global optimum solution, the branching strategy plays a central role in determining the performance of the algorithm. A judicious choice of branching variable may result in the elimination of large subdomains of the solution space. This choice also affects the quality of the underestimating problem both directly, as shown in equation (2), and indirectly, through the calculation of α.

Most mixed–integer optimization algorithms rely on a branching strategy that involves the binary variables exclusively. Although this approach helps to address the combinatorial nature of the problem at hand, it does not resolve the nonconvexity issues. When all the binary variables have been fixed, the resulting NLP and its convex lower bounding problem are in general too distant to decide conclusively whether the global optimum point lies in the region corresponding to this 0–1 combination. Thus, the branching strategy used in the SMIN–αBB algorithm is based on a hybrid scheme which involves both continuous and binary variables. Branching on the continuous variables is aimed at improving the quality of the underestimators and therefore utilizes the set of branching rules devised for the αBB algorithm.

Two different approaches have been implemented in order to determine whether branching for a specific region should occur on a binary or a continuous variable, and which variable should be selected. Note that two subregions (nodes) are created during each branching step or iteration.

Binary Variables First In this case, the m binary variables indicated by the user as branching variables are given priority over the continuous variables. As a result, the generation of lower bounds at all the nodes on the first $(m-1)$ levels of the branch–and–bound tree requires the solution of convex MINLPs but for subsequent levels, all the binary variables are fixed and convex NLPs must then be solved. The choice of an appropriate binary variable can be based either on the solution of a continuous relaxation of the nonconvex MINLP at the current node or on a random selection.

Least Fractional First With this option, binary and continuous branching can occur at any level of the branch–and–bound tree. If the set of binary branching variables is not empty at the current node, a continuous relaxation of the nonconvex MINLP is solved locally. If the optimal value of one of the relaxed variables is within a user–specified distance, *ydist*, of 0 or 1, this variable becomes a candidate for branching. If no binary candidates have been selected after this process, branching occurs on one of the continuous variables.

Variable Bound Updates Strategy

Since the size of the continuous domain greatly affects the quality of the underestimating problems, the solution space should be reduced as much as possible before these problems are constructed. Several strategies have been devised for this purpose within the αBB algorithm (Adjiman et al., 1997b) and they are also used for the SMIN–αBB. In addition, it is desirable to update the bounds on the binary variables as this allows the elimination of some binary combinations. A strategy which relies on interval analysis (Neumaier, 1990) has therefore been developed. After a binary variable has been chosen, its bounds are set to 0. Using interval arithmetic and the current variable bounds, the constraints are then evaluated. Note that all binary variables other than the selected variable are treated as continuous variables with 0–1 bounds during this step. If at least one constraint is deemed infeasible, it is known with certainty that the selected binary variable cannot take on a value of 0. The same procedure is repeated with the binary variable bounds set to 1. If both 0 and 1 are infeasible, the entire region can be fathomed. This approach to bound updates is computationally inexpensive compared to that used for continuous variables : only interval evaluations are required, whereas several convex optimization problems are needed for the continuous variables. Yet, it constitutes a very effective scheme to alleviate the combinatorial problems posed by the presence of binary variables.

Computational Results

Small Examples Four small examples were taken from the literature. All the binary variables participate linearly and separably in the problems, but the continuous terms exhibit different types of nonconvexities. The results are shown in Table 2. The number of continuous and binary variables are denoted by nx and ny respectively. Problem 1 ($nx = 7, ny = 2$) corresponds to Example 1 in Kocis and Grossmann (1989). Problem 2 ($nx = 2, ny = 3$) is Example 6 from Floudas et al. (1989). Problem 3 ($nx = 2, ny = 1$) is taken from Floudas (1995) and corresponds to Example 6.6.5. Finally, Problem 4 ($nx = 3, ny = 6$) is a quadratic constrained problem. All runs were performed on an HP–C160, branching on the binary variables first and updating both binary and continuous variables.

Heat Exchanger Network Problem This heat exchanger network optimization problem is taken from Yee and Grossmann (1991). It involves two hot process streams, two cold process streams, a cold and a hot utility. There are two temperature intervals and therefore 12 binary variables representing potential matches. All the constraints are linear but the problem is posed as a cost minimization so that the objective function is highly nonlinear. A brute force approach would therefore require the solution of 4096 nonconvex NLPs. A set of preliminary runs are presented in Figure 1 and Table 1. The “deepest level” column lists the furthest level of the branch and bound tree reached during the exploration of the solution space. The “binary branches” column keeps track of the number of times binary variables are selected for branching out of a maximum of 4095 instances. The importance of the branching strategy is clearly demonstrated by these results : without branching or bound updates on the binary variables (Run 1), the convergence rate soon becomes asymptotic and the run is not completed after 800 iterations. Different approaches for the choice of branching variable and variable bound updates lead to markedly different performances. In Run 2, bound updates are performed exclusively on the continuous variables but branching occurs on the binary variables first. In Run 3, the same branching procedure is used but bound updates are also used on the binary variables. In Run 4, branching on the binary variables is now based on the solution of a relaxed NLP with $ydist$ set to 0.1. In Run 5, $ydist$ is equal to 0.2. Finally, Run 6 is similar to Run 5 except that a smaller number of continuous variables are selected for bound updates at each iteration. Although the smallest number of iterations is achieved when using the "least fractional first" branching strategy (Run 5), the smallest CPU corresponds to the "binary variables first" option (Run 3). Because all the binary variables are eligible for branching in Run 3, 705 nodes involve the solution of a convex MINLP, and the remaining 504 nodes that of a convex NLP. In Run 5, the binary variables have to meet an additional criterion to be branched on so that time–consuming convex MINLPs have to be solved at 789 out of a total 845 nodes.

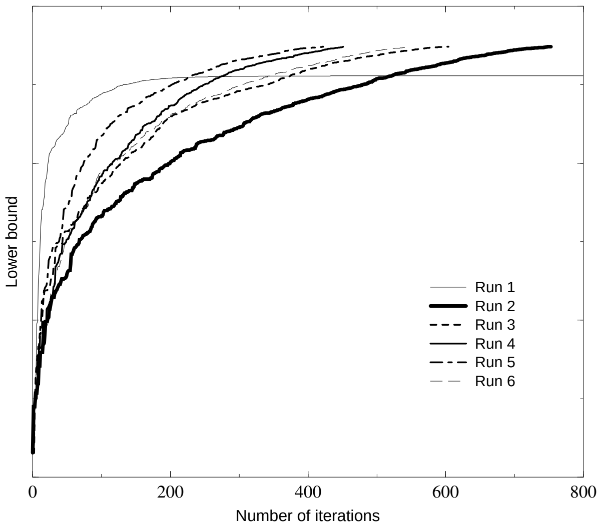

Figure 1 – Lower bound progress for heat exchanger network

Run	Iterations	CPU sec	Deepest level	Binary branches
1	800	2210	60	—
2	753	1116	26	343
3	604	755	23	173
4	451	1041	18	97
5	422	935	26	112
6	547	945	22	127

Table 1 - Optimization of a heat exchanger network
Note that Run 1 converges asymptotically

THE GMIN–αBB ALGORITHM

The GMIN–αBB algorithm is a global optimization algorithm for general MINLPs of the form :

$$\begin{array}{lrl} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x},\mathbf{y}) & \\ s.t. & \mathbf{g}(\mathbf{x},\mathbf{y}) & \leq 0 \\ & \mathbf{h}(\mathbf{x},\mathbf{y}) & = 0 \\ & \mathbf{x} \in X \subseteq \Re^n & \\ & \mathbf{y} \in Y = \{0,1\}^m & \end{array} \tag{3}$$

where f, $\mathbf{g}$ and $\mathbf{h}$ belong to $\mathcal{C}^2$, the set of twice-differentiable functions, $\mathbf{x}$ is a vector of size n, $\mathbf{y}$ is a vector of size m.

As seen from the formulation, arbitrary twice–continuous non–convex terms in which both continuous and binary variables participate are allowed. Thus, the proposed framework addresses a very general class of MINLP problems.

The key idea of the GMIN–αBB algorithm is to embed the αBB algorithm within a branch and bound framework which handles the binary variables. At each node of the branch and bound tree a continuous relaxation of the original problem is being solved, with some of the binary variables fixed to 0 or 1, according to the branching rules that are discussed in a subsequent section. The most important consequence of this approach is that the continuous relaxation at each node is a *non–convex* NLP whose *global optimum solution* can provide a guaranteed lower bound to the MINLP problem. It is therefore crucial to be able to solve each branch and bound node efficiently to global optimality. Note that any lower bound on the global solution of these non–convex NLPs is a valid lower bound for the global solution of the original MINLP problem. The αBB algorithm is employed so as to provide valid lower bounds for the outer branch and bound algorithm.

Three major issues have to be addressed to devise an effective branch and bound algorithm, namely lower bound generation, selection of branching variables, selection of branching node. These are general features of any such scheme but their importance is accentuated when one is faced with a difficult problem such as the global optimization of general non-convex MINLP problems. In the sequel, some key ideas related to the aforementioned issues are addressed together with their implications in the design of our branch and bound global optimization algorithm.

Lower Bound Generation

Branch and bound is a very broad concept that defines a generic framework for addressing a large variety of optimization problems. What sets apart one implementation of a branch and bound algorithm from another is the way in which the necessary information for each node is obtained. Nodes are usually characterized by a *lower* bound, i.e. a value that bounds the solution of the problem from below, within the given region. In practice, this bound corresponds to a relaxation of the original problem or equivalently to an underestimation of the problem. In order to guarantee global optimality, this underestimation must be valid at all points within the specified region.

Within our framework, the lower bound is defined as the global solution of the NLP relaxation of the MINLP at a selected node, where some of the binary variables are set to their bounds. The fact that the resulting NLP problem is *nonconvex* implies that the validity of the underestimate of the MINLP for the current node can only be guaranteed if a global optimization algorithm for twice–differentiable continuous problems such as the αBB is used.

Preserving all guarantees of global optimality, a valid lower bound for the MINLP can also be obtained by generating a valid lower bound for the nonconvex NLP. One of the many advantages of αBB algorithm is that it identifies the global minimum of an NLP by generating a valid lower bound at every iteration. This implies that one may wish to interrupt the search for the global minimum of particular node before convergence is achieved and use the lower bound of αBB algorithm as the lower bound to the MINLP. In this manner, the high computational cost sometimes incurred when solving to ϵ–global optimality NLP problems can be overcome, while at the same time valid lower bounds for the MINLP are generated. Finally, in order to compute upper bounds to the MINLP problem, a local solution of the continuous relaxation is obtained and is kept if it is integer feasible, that is, if it satisfies the nonconvex MINLP.

Selection of Branching Variables

The ability to identify the path along a branch and bound tree that would result in the exploration of the smallest possible number of nodes is of paramount importance to ensure the applicability of the algorithm. This is to a great extent determined by the particular combination of fixed and relaxed binary variables that is selected for exploration. Ideally one would like to identify those variables that have the most pronounced effect on the problem structure. Over the years, researchers have experimented extensively with rules to be used to decide on which variable to branch so as to generate new nodes (Gupta and Ravindran, 1985; Sherali and Myers, 1985). Among the developed procedures, the most successful ones correspond to a strategy that selects to branch on the most fractional variables and one that uses knowledge about a problem in order to assign different branching priorities to the binary variables. In our examples, a hybrid of the two will be discussed and the major advantages will be presented.

Selection of Branching Nodes

The path the search follows is also greatly affected by which node is being selected for further branching. The node that generated the *least* lower bound and *newest* node generated are two potential candidates. The first alternative represents an intuitive rule which allows the exploration of the most promising region based on the quality of the lower bounds.

Computational Results

Small Examples The same test problems were used as for the SMIN–αBB algorithm and the results are reported in Table 2.

Problem	SMIN–αBB		GMIN–αBB	
	Iter.	CPU sec.	Iter.	CPU sec.
1	9	6.70	2	1.19
2	4	0.40	1	0.14
3	9	0.53	2	0.06
4	2	0.47	8	0.73

Table 2 – Computational results for the small examples

Design of Pump Configurations This problem, discussed in Westerlund et al. (1994), aims to identify the configuration of a system of centrifugal pumps so as to achieve a pre–specified pressure drop while the total flow rate is specified. It is defined via a non–linear mixed integer optimization problem. There are up to three levels. The discrete decisions are as follows : $z_i = 0$ if a particular level does not exist, 1 otherwise; Np_i is the number of parallel lines at level i ($Np_i \leq 3$); Ns_i is the number of pumps in series at level i ($Ns_i \leq 3$). The last two variables are integer variables. Therefore their binary expansion is used as $Np_i = yp_{i,1} + 2yp_{i,2}$, and $Ns_i = ys_{i,1} + 2ys_{i,2}$. In all cases $i = 1, 2, 3$. The formulation used therefore involves 15 binary variables. Thirty seven local minima have previously been reported for this problem.

$$
\begin{aligned}
\min_{Np_i, Ns_i, z_i, w_i, x_i} \quad & \sum_{i=1}^{3} (C_i + C_i\prime P_i) Np_i Ns_i z_i \\
s.t. \quad & P_1 - 19.9(\tfrac{\omega_1}{2950})^3 - 0.1610(\tfrac{\omega_1}{2950})^2 \dot{v}_1 \\
& \qquad + 0.000561(\tfrac{\omega_1}{2950})\dot{v}_1{}^2 = 0 \\
& P_2 - 1.21(\tfrac{\omega_2}{2950})^3 - 0.0644(\tfrac{\omega_2}{2950})^2 \dot{v}_2 \\
& \qquad + 0.000564(\tfrac{\omega_2}{2950})\dot{v}_2{}^2 = 0 \\
& P_3 - 6.52(\tfrac{\omega_3}{2950})^3 - 0.1020(\tfrac{\omega_3}{2950})^2 \dot{v}_3 \\
& \qquad + 0.000232(\tfrac{\omega_3}{2950})\dot{v}_3{}^2 = 0 \\
& \Delta p_1 - 629(\tfrac{\omega_1}{2950})^2 + 0.696(\tfrac{\omega_1}{2950})\dot{v}_1 \\
& \qquad - 0.0116\dot{v}_1{}^2 = 0 \\
& \Delta p_2 - 215(\tfrac{\omega_2}{2950})^2 + 2.950(\tfrac{\omega_2}{2950})\dot{v}_2 \\
& \qquad - 0.0115\dot{v}_2{}^2 = 0 \\
& \Delta p_3 - 361(\tfrac{\omega_3}{2950})^2 + 0.530(\tfrac{\omega_3}{2950})\dot{v}_3 \\
& \qquad - 0.0115\dot{v}_3{}^2 = 0 \\
& x_1 + x_2 + x_3 = 1 \\
& \omega_i - 2950 \leq 0 \\
& \dot{v}_i - \tfrac{x_i}{Np_i} V_{tot} = 0 \\
& \Delta P_{tot} - \Delta p_i Ns_i = 0 \\
& V_{tot} = 350, \Delta P_{tot} = 400 \\
& Np_i = \{1, 2, 3\}, Ns_i = \{1, 2, 3\}, i = 1, 2, 3 \\
& z_i = \{0, 1\}, i = 1, 2, 3
\end{aligned}
\tag{4}
$$

Because the relaxed NLP formulation is highly nonlinear, the αBB algorithm was not run to completion when the lower bounding problem for a node was solved in order to improve the computational efficiency. Furthermore, certain binary variables are more relevant than others in terms of their contribution to the complexity of the problem. Specifically, the binary variables, z_i, associated with the existence or non–existence of a level are important because they help to reduce the size of the NLP problem by setting the variables associated with a particular level to zero when that level does not exist. This is achieved by augmenting formulation (4) with a set of constraints of the form:

$$d_i - z_i \leq 0$$

$$d_i = \{P_i, \Delta P_i, \dot{v}_i, \omega_i, x_i, \frac{Np_i}{Np^{max}}, \frac{Ns_i}{Ns^{max}}\}$$

$$Ns_i - Np^{max} Np_i \leq 0$$

If a particular combination of the z_i variables is excluded early on, a substantial portion of the binary branch and bound tree can immediately be fathomed. This leads us to a hybrid branching scheme in which high priorities are assigned to the three binary variables defining the existence or non-existence of a level. Eight nodes of the branch and bound tree are defined (all possible combinations of the three binary variables) and these 8 problems are solved first. This way, one immediately identifies that in the absence of levels 1 and 3 the problem is infeasible. As a result a sizeable portion of the branch and bound tree can be excluded. Once the first lower bounds have been generated, the strategy that assigns the highest priority to the most fractional binary variable is employed. The global minimum configuration is identified after exploring 162 nodes out of the 2^{15} total nodes.

CONCLUSIONS

Two branch–and–bound algorithms for the global optimization of MINLPs with twice–differentiable functions in the continuous variables were presented in this paper. The type of participation allowed for the binary variables differs for the two algorithms : in the SMIN–αBB, valid lower bounds are obtained by constructing and solving a convex MINLP in which the binary variables participate in linear or mixed–bilinear terms; in the GMIN–αBB, a continuous relaxation is solved to global optimality or rigorously underestimated and more complex binary or mixed terms can therefore be handled. Both approaches use recent developments for the valid underestimation of general functions with continuous second–order derivatives. A variety of branching and variable bound updates strategies have been studied for both algorithms and they have been shown to greatly affect their performance. A few literature problems and two more realistic examples have been used successfully to illustrate potential applications of these global optimization algorithms for a broad class of problems.

ACKNOWLEDGMENTS

Financial support from the National Science Foundation and Mobil is gratefully acknowledged.

REFERENCES

Adjiman C.S., Androulakis I.P., and Floudas C.A., 1997b, A global optimization method, αBB, for general twice–differentiable NLPs – II. Implementation and computational results. in preparation.

Adjiman C.S., Androulakis I.P., Maranas C.D., and Floudas C.A., 1996, A global optimisation method, αBB, for process design. *Computers chem. Engng, Suppl.* **20**, S419–S424.

Adjiman C.S., Dallwig S., Floudas C.A., and Neumaier A., 1997a, A global optimization method, αBB, for general twice–differentiable NLPs – I. Theoretical advances. in preparation.

Adjiman C.S. and Floudas C.A., 1996, Rigorous convex underestimators for general twice–differentiable problems. *J. of Glob. Opt.* **9**, 23–40.

Androulakis I., Maranas C.D., and Floudas C.A., 1995, αBB : A global optimization method for general constrained nonconvex problems. *J. of Glob. Opt.* **7**, 337–363.

Duran M.A. and Grossmann I.E., 1986, An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Math. Prog.* **36**, 307.

Fletcher R. and Leyffer S., 1994, Solving mixed integer nonlinear programs by outer approximation. *Math. Prog.* **66**, 327.

Floudas C., Aggarwal A., and Ciric A., 1989, Global optimal search for nonconvex NLP and MINLP problems. *Comp. Chem. Engng.* **13**, 1117.

Floudas C.A., 1995, *Nonlinear and Mixed Integer Optimization : Fundamentals and Applications*. Oxford University Press, New York.

Floudas C.A. and Grossmann I.E., 1995, Algorithmic approaches to process synthesis : Logic and global optimization. In *FOCAPD'94*, vol. 91 of *AICHE Symposium Series*, (pp. 198–221).

Geoffrion A.M., 1972, Generalized Benders decomposition. *J. of Opt. Th. and Appl.* **10**, 237–260.

Glover F., 1975, Improved linear integer programming formulations of nonlinear integer problems. *Mgmt. Sci.* **22**, 445.

Grossmann I.E., 1990, MINLP optimization strategies and algorithms for process synthesis. In J. J. Siirola I.E.G. and Stephanopoulos G. (eds.), *Proceedings of FOCAPD'90*, Cache-Elsevier, Amsterdam.

Grossmann I.E., 1996, Mixed–integer optimization techniques for algorithmic process synthesis. In Anderson J. (ed.), *Adv. in Chem. Eng.*, vol. 23, (pp. 172–239).

Gupta O.K. and Ravindran R., 1985, Branch and bound experiments in convex nonlinear integer programing. *Mgmnt. Sci.* **31**, 1533–1546.

Holmberg K., 1990, On the convergence of the cross decomposition. *Math. Prog.* **47**, 269.

Kocis G.R. and Grossmann I.E., 1987, Relaxation strategy for the structural optimization of process flowsheets. *I & EC Res.* **26**, 1869.

Kocis G.R. and Grossmann I.E., 1989, A modeling and decomposition strategy for the MINLP optimization of process flowsheets. *Comp. Chem. Engng.* **13**, 797.

Maranas C.D. and Floudas C., 1993, Global optimization for molecular conformation problems. *Annals of Op. Res.* **42**, 85–117.

Neumaier A., 1990, *Interval Methods for Systems of Equations*. Encyclopedia of Mathematics and its Applications, Cambridge University Press.

Ryoo H.S. and Sahinidis N.V., 1995, Global optimization of nonconvex nlps and minlps with applications in process design. *Computers chem. Engng* **19**, 551–566.

Sherali H. and Myers D., 1985, The design of branch and bound algorithms for a class of nonlinear integer problems. *Annals of Op. Res.* **5**, 463–484.

Viswanathan J. and Grossmann I.E., 1990, A combined penalty function and outer-approximation for MINLP optimization. *Comp. Chem. Engng.* **14**, 769–782.

Westerlund T. and Pettersson F., 1995, An extended cutting plane method for solving convex MINLP problems. *Computers chem. Engng, Suppl.* **19**, 131–136.

Westerlund T., Pettersson F., and Grossmann I.E., 1994, Optimization of pump configuration problems as a MINLP probem. *Comp. Chem. Engng.* **18**, 845–858.

Yee T.F. and Grossmann I.E., 1991, Simultaneous optimization model for heat exchanger network synthesis. In Grossmann I.E. (ed.), *Chemical Engineering Optimization Models with GAMS*, vol. 6 of *CACHE Design Case Studies Series*.