DIMACS Series in Discrete Mathematics
and Theoretical Computer Science

Nonlinear and Mixed-Integer Optimization in Chemical Process Network Systems

C. S. Adjiman, C. A. Schweiger, and C. A. Floudas*

ABSTRACT. The use of networks allows the representation of a variety of important engineering problems. The treatment of a particular class of network applications, the *Process Synthesis problem*, is exposed in this paper. Process Synthesis seeks to develop systematically process flowsheets that convert raw materials into desired products. In recent years, the optimization approach to process synthesis has shown promise in tackling this challenge. It requires the development of a network of interconnected units, the process superstructure, that represents the alternative process flowsheets. The mathematical modeling of the superstructure has a mixed set of binary and continuous variables and results in a mixed-integer optimization model. Due to the nonlinearity of chemical models, these problems are generally classified as Mixed-Integer Nonlinear Programming (MINLP) problems.

A number of local optimization algorithms for MINLP problems are outlined in this paper: Generalized Benders Decomposition (GBD), Outer Approximation (OA), Generalized Cross Decomposition (GCD), Extended Cutting Plane (ECP), Branch and Bound (BB), and Feasibility Approach (FA), with particular emphasis on the Generalized Benders Decomposition. Recent developments for the global optimization of nonconvex MINLPs are then introduced. In particular, two branch-and-bound approaches are discussed: the Special structure Mixed Integer Nonlinear αBB (SMIN-αBB), where the binary variables should participate linearly or in mixed-bilinear terms, and the General structure Mixed Integer Nonlinear αBB (GMIN-αBB), where the continuous relaxation of the binary variables must lead to a twice-differentiable problem. Both algorithms are based on the αBB global optimization algorithm for nonconvex continuous problems.

Once some of the theoretical issues behind local and global optimization algorithms for MINLPs have been exposed, attention is directed to their practical use. The algorithmic framework MINOPT is discussed as a computational tool for the solution of process synthesis problems. It is an implementation of a number of local optimization algorithms for the solution of MINLPs. The synthesis problem for a heat exchanger network is then presented to demonstrate the application of some local MINLP algorithms and the global optimization SMIN-αBB algorithm.

1991 *Mathematics Subject Classification.* Primary 54C40, 14E20; Secondary 46E25, 20C20.

This work was supported by the National Science Foundation and Mobil Technology Company.

*Author to whom all correspondence should be addressed.

©0000 (copyright holder)

1. Introduction

Network applications exist in many fields including engineering, applied mathematics, and operations research. These applications include problems such as facility location and allocation problems, design and scheduling of batch processes, facility planning and scheduling, topology of transportation networks, and process synthesis problems. These types of problems are typically characterized by both discrete and continuous decisions. Thus, the modeling aspects of these applications often lead to models involving both integer and continuous variables as well as nonlinear functions. This gives rise to problems classified as mixed-integer nonlinear optimization problems.

Major advances have been made in the development of mathematical programming approaches which address mixed-integer nonlinear optimization problems. The recent theoretical and algorithmic advances in mixed-integer nonlinear optimization have made the use of these techniques both feasible and practical. Because of this, optimization has become a standard computational approach for the solution of these networking problems.

Some of the major contributions to the development of mixed-integer nonlinear optimization techniques have come from the field of process synthesis. This is due to the natural formulation of the process synthesis problem as a mixed-integer nonlinear optimization problem. This has led to significant algorithmic developments and extensive computational experience in process synthesis applications. The research in this area has focused on the overall process synthesis problem as well as subsystem synthesis problems including heat exchanger network synthesis (HENS), reactor network synthesis, distillation sequencing, and mass exchange network synthesis.

The process synthesis problem is stated as follows: given the specifications of the inputs (feed streams) and the specifications of the outputs, develop a process flowsheet which transforms the given inputs to the desired products while addressing the performance criteria of capital and operating costs, product quality, environmental issues, safety, and operability. Three key issues must be addressed in order to determine the process flowsheet: which process units should be in the flowsheet, how the process units should be interconnected, and what the operating conditions and sizes of the process units should be. The optimization approach to process synthesis has been developed to address these issues and has led to some of the major theoretical and algorithmic advances in mixed-integer nonlinear optimization.

The next section describes the optimization approach to process synthesis which leads to the formulation of a Mixed-Integer Nonlinear Program. In Section 3, the Generalized Benders Decomposition, one of the optimization algorithms developed for the solution of the posed optimization problem, is presented. Although this and other MINLP algorithms have been developed for process synthesis, they are applicable to models that result in other network applications. Section 4 reports some recent developments for the global optimization of nonconvex MINLPs. Section 5 describes the algorithmic framework, **MINOPT**, which implements a number of MINLP algorithms. The final part of the paper describes the application of both global and local MINLP methods to a heat exchanger network synthesis problem.

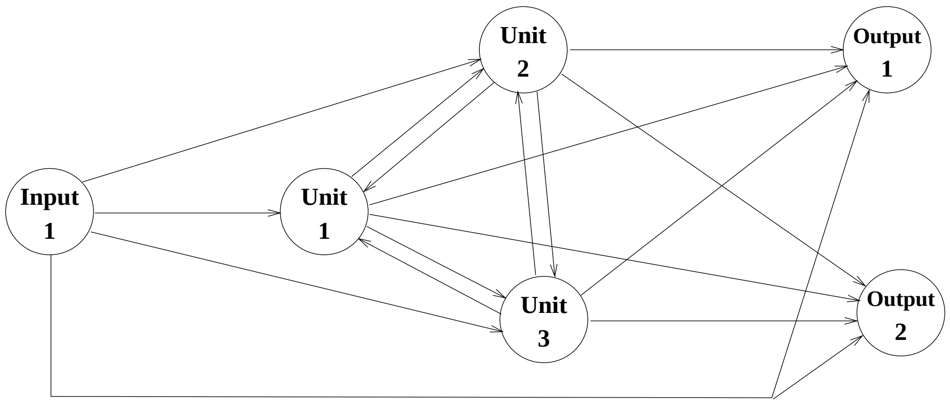

FIGURE 1. Network representation of superstructure

2. Optimization Approach in Process Synthesis

A major advance in process synthesis has been the development of the optimization approach. This approach leads to a mathematical programming problem classified as a Mixed Integer Nonlinear Program. Significant progress has been made in the development of algorithms capable of addressing this class of problems.

The optimization approach to process synthesis involves three steps: the representation of alternatives through a process superstructure, the mathematical modeling of the superstructure, and the development of an algorithm for the solution of the mathematical model. Each of these steps is crucial to the determination of the optimal process flowsheet.

The superstructure is a superset of all process design alternatives of interest. The representation of process alternatives is conceptually based on elementary graph theory ideas. Nodes are used to represent the inputs, outputs, and each unit in the superstructure. One-way arcs represent connections from inputs to process units, two-way arcs represent interconnections between process units, and one-way arcs represent connections to the outputs. The result is a bi-partite planar graph which represents the network of process units in the superstructure. This network represents all the options of the superstructure and includes cases where nodes in the graph may or may not be present. The idea of the process superstructure can be illustrated by a process which has one input, two outputs, and potentially three process units. The network representation of this is shown in Figure 1.

Since all the possible candidates for the optimal process flowsheet are embedded within this superstructure, the optimal process flowsheet that can be determined is only as good as the postulated representation of alternatives. This superstructure must be rich enough to allow for a complete set of alternatives, but it must also be concise enough to eliminate undesirable structures.

A specific example of a superstructure is illustrated by the two component distillation scheme presented by [**KG89**]. This process consists of two feed streams of known composition and flowrate and two products streams with specified purities. The superstructure consists of a flash unit and a distillation unit and is shown in Figure 2.

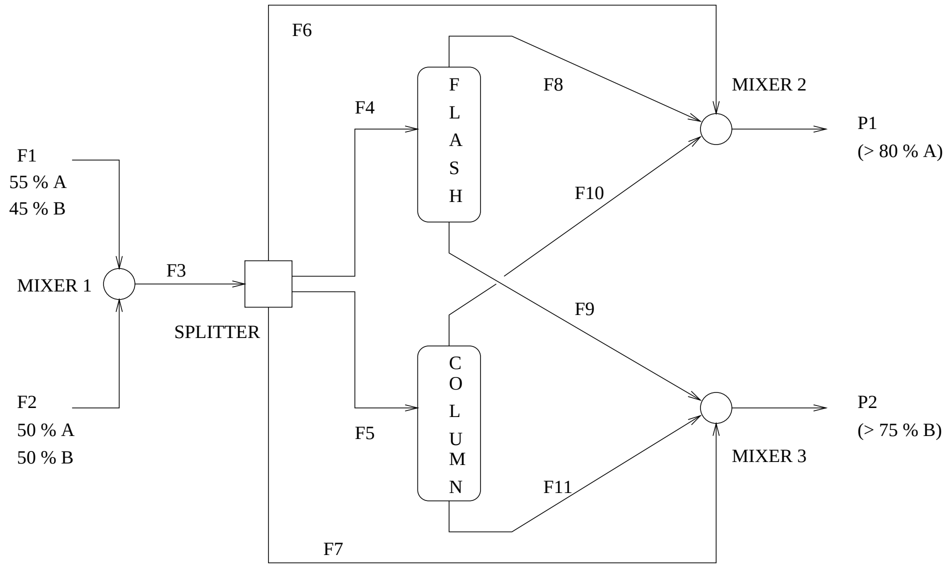

FIGURE 2. A Two-Column Distillation Sequence Superstructure

Through the process synthesis, the structure flowsheet and the optimal values of the operating parameters are determined. The existence of process units leads to discrete decisions while the determination of operating parameters leads to continuous decisions. Thus, the process synthesis problem is mathematically classified as mixed discrete-continuous optimization.

The next step involves the mathematical modeling of the superstructure. Binary variables are used to indicate the existence of nodes within the network and continuous variables represent the levels of values along the arcs. The resulting formulation is a Mixed Integer Nonlinear Programming Problem (MINLP):

$$\begin{array}{rrcl} \min\limits_{\boldsymbol{x},\boldsymbol{y}} & f(\boldsymbol{x},\boldsymbol{y}) & & \\ \text{s.t.} & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) & = & \mathbf{0} \\ & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) & \leq & \mathbf{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} & \in & \boldsymbol{Y} \quad \text{integer} \end{array} \tag{2.1}$$

where

- $\boldsymbol{x}$ is a vector of n continuous variables representing flow rates, compositions, temperatures, and pressures of process streams and sizing of process units.
- $\boldsymbol{y}$ is a vector of integer variables representing process alternatives.
- $f(\boldsymbol{x},\boldsymbol{y})$ is the single objective function representing the performance criterion.
- $\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = \mathbf{0}$ are the m equality constraints that represent the mass and energy balances, and equilibrium expressions.
- $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq \mathbf{0}$ are the p inequality constraints that represent design specifications, restrictions, and logical constraints.

This formulation is completely general and includes cases where nonlinearities occur in the $\boldsymbol{x}$ space, $\boldsymbol{y}$ space, and joint $\boldsymbol{x}-\boldsymbol{y}$ space.

The integer variables can be expressed as binary variables without loss of generality. Through an appropriate transformation, the general formulation can be written as

$$
\begin{array}{rrcl}
\min\limits_{\boldsymbol{x},\boldsymbol{y}} & f(\boldsymbol{x},\boldsymbol{y}) & & \\
\text{s.t.} & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) & = & \mathbf{0} \\
& \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) & \leq & \mathbf{0} \\
& \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{y} & \in & \boldsymbol{Y} = \{0,1\}^q
\end{array}
\tag{2.2}
$$

where the $\boldsymbol{y}$ are the q binary variables which represent the existence of process units.

The final step of the optimization approach is the development and application of algorithms for the solution of the mathematical model. This step is highly dependent on the properties of the mathematical model and makes use of the structure of the formulation. This step focuses on the development of algorithms capable of addressing the MINLPs.

The solution of MINLPs is particularly challenging due to the combinatorial nature of the problem ($\boldsymbol{y}$ domain) combined with the nonlinearities in the continuous domain ($\boldsymbol{x}$ domain). The combinatorial nature of the problem becomes an issue as the number of $\boldsymbol{y}$ variables increases creating a large number of possible process structures. In the continuous domain, the models of chemical processes are generally nonlinear. The nonlinearities in the problem imply the possible existence of multiple solutions and lead to challenges in finding the global solution.

Despite the challenges involved in the solution of the MINLPs, there have been significant advances in the area of MINLPs on the theoretical, algorithmic, and computational fronts. Many algorithms have been developed to address problems with the above form and a brief review of these developments with emphasis on the Generalized Benders Decomposition is presented in the next section.

3. MINLP Algorithms

A number of algorithms have been developed to address the problem formulation 2.2. The following is a listing of these algorithms.

1. Generalized Benders Decomposition, **GBD** [**Geo72, PF89, FAC89**]
2. Branch and Bound, **BB** [**Bea77, Gup80, OOM90, BM91, QG92**]
3. Outer Approximation, **OA** [**DG86**]
4. Feasibility Approach, **FA** [**MM86**]
5. Outer Approximation with Equality Relaxation, **OA/ER** [**KG87**]
6. Outer Approximation with Equality Relaxation and Augmented Penalty, **OA/ER/AP** [**VG90**]
7. Generalized Outer Approximation, **GOA** [**FL94**]
8. Generalized Cross Decomposition, **GCD** [**Hol90**]
9. Extended Cutting Plane, **ECP** [**WPG94, WP95**]
10. Logic Based Methods, [**RG94, TG96**]
11. Interval Analysis Based Methods, [**VEH96**]

An overview of MINLP algorithms and extensive theoretical, algorithmic, and applications-oriented description of **GBD**, **OA**, **OA/ER**, **OA/ER/AP**, **GOA**, and **GCD** algorithms is found in [**Flo95**].

Some of these algorithms are applicable only to restricted classes of the general problem formulation. The general strategy of algorithms used to solve MINLPs is to formulate subproblems such that the subproblems are easier to solve than the original problem. This may involve fixing certain variable types, relaxing certain constraints, using duality, or using linearization. The algorithms iterate through solutions of the subproblems which provide upper and lower bounds on the optimal solution of the original problem. The nature of the subproblems and the quality of bounds provided by the subproblems are different for the various algorithms.

Due to space limitations, only one of these algorithms, Generalized Benders Decomposition, is discussed here. The work of [**Geo72**] generalized the work of [**Ben62**] which exploits the structure of mathematical programming problems. The algorithm addresses problems with the form of problem 2.2. In fact, the algorithm is applicable to a broader class of problems for which the $\boldsymbol{y}$ variables may be continuous. The focus here is on MINLP models and thus the $\boldsymbol{y}$ variables will be treated as binary.

The basic idea behind **GBD** is the generation of upper and lower bounds on the solution of the MINLP model through the iterative solution subproblems formulated from the original problem. The upper bound is the result of the solution of the *primal* problem while the lower bound is the result of the solution of the *master* problem. The primal problem corresponds to the solution of the original problem 2.2 with the values of the $\boldsymbol{y}$ variables fixed. This problem is solved in the $\boldsymbol{x}$ space only and its solution provides information about the Lagrange multipliers for the constraints. The master problem is formulated by making use of the Lagrange multipliers and nonlinear duality theory. Its solution provides a lower bound as well as a new set of $\boldsymbol{y}$ variables. The algorithm iterates between the primal and master problems generating a sequence of upper and lower bounds which converge in a finite number of iterations.

3.1. Primal Problem. The primal problem results from fixing the values of the $\boldsymbol{y}$ variables. For values of $\boldsymbol{y}$ fixed to $\boldsymbol{y}^k$ where k is an iteration counter, the primal problem has the following formulation:

$$
\begin{array}{rrcl}
\min\limits_{\boldsymbol{x}} & f(\boldsymbol{x},\boldsymbol{y}^k) & & \\
\text{s.t.} & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}^k) & = & \mathbf{0} \\
& \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}^k) & \leq & \mathbf{0} \\
& \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n
\end{array} \tag{3.1}
$$

The primal formulation is an NLP which can be solved by using existing algorithms. If the primal problem is feasible, then the optimal solution provides values for $\boldsymbol{x}^k$, $f(\boldsymbol{x}^k,\boldsymbol{y}^k)$, and the Lagrange multipliers λ^k and μ^k for the equality and inequality constraints.

If the primal problem is found to be infeasible when applying a solution algorithm, a feasibility problem is formulated. This problem can be formulated by minimizing the ℓ_1 or ℓ_∞ sum of constraint violations. One possible formulation of the feasibility problem is the following:

$$
\begin{array}{llrcl}
 & \min\limits_{\boldsymbol{x},\alpha} & \alpha_i + \alpha_e^+ + \alpha_e^- & & \\
 & \text{s.t.} & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}^k) - \alpha_i & \leq & \boldsymbol{0} \\
(3.2) & & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}^k) + \alpha_e^+ - \alpha_e^- & = & \boldsymbol{0} \\
 & & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
 & & \alpha_i, \alpha_e^+, \alpha_e^- & \geq & 0
\end{array}
$$

Another possible form for the infeasible primal problem is the following where the equality constraints are not relaxed:

$$
\begin{array}{lll}
 & \min\limits_{\boldsymbol{x},\alpha} & \alpha \\
 & \text{s.t.} & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}^k) \leq \alpha \\
(3.3) & & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}^k) = \boldsymbol{0} \\
 & & \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \\
 & & \alpha \geq 0
\end{array}
$$

The solution of the feasibility problem provides values for $\bar{\boldsymbol{x}}^k$ and the Lagrange multipliers $\bar{\lambda}^k$ and $\bar{\mu}^k$ for the equality and inequality constraints.

3.2. Master Problem. The formulation of the master problem for **GBD** makes use of nonlinear duality theory. The key aspects of the master problem formulation are the projection of the problem onto the $\boldsymbol{y}$ space and the dual representation.

For the projection of the problem onto the $\boldsymbol{y}$ space, problem 2.2 can be written as

$$
\begin{array}{llrcl}
 & \min\limits_{\boldsymbol{y}} \inf\limits_{\boldsymbol{x}} & f(\boldsymbol{x},\boldsymbol{y}) & & \\
 & \text{s.t.} & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) & = & \boldsymbol{0} \\
(3.4) & & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) & \leq & \boldsymbol{0} \\
 & & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
 & & \boldsymbol{y} & \in & \boldsymbol{Y} \equiv \{0,1\}^q
\end{array}
$$

Let $v(\boldsymbol{y})$ and $\boldsymbol{V}$ be defined as follows:

$$
\begin{array}{llrcl}
 & v(\boldsymbol{y}) = \inf\limits_{\boldsymbol{x}} & f(\boldsymbol{x},\boldsymbol{y}) & & \\
(3.5) & \text{s.t.} & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) & = & \boldsymbol{0} \\
 & & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) & \leq & \boldsymbol{0} \\
 & & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n
\end{array}
$$

$$
(3.6) \qquad \boldsymbol{V} = \{\boldsymbol{y} : \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{0}, \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq \boldsymbol{0} \quad \text{for some} \quad \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n\}
$$

The projected problem can now be written as:

$$
\begin{array}{lll}
 & \min\limits_{\boldsymbol{y}} & v(\boldsymbol{y}) \\
(3.7) & \text{s.t.} & \boldsymbol{y} \in \boldsymbol{Y} \cap \boldsymbol{V}
\end{array}
$$

The difficulty in solving this problem is that $\boldsymbol{V}$ and $v(\boldsymbol{y})$ are known only implicitly. In order to overcome this, dual representations of $\boldsymbol{V}$ and $v(\boldsymbol{y})$ are used.

The dual representation of $\boldsymbol{V}$ is described in terms of a collection of regions that contain it. An element of $\boldsymbol{Y}$ also belongs to the set $\boldsymbol{V}$ if and only if it satisfies the system:

$$\begin{array}{rrcl} & \mathbf{0} & \geq & \inf \bar{L}(\boldsymbol{x}, \boldsymbol{y}, \bar{\lambda}, \bar{\mu}), \quad \forall \bar{\lambda}, \bar{\mu} \in \Lambda \\ \text{where} & \Lambda & = & \left\{ \bar{\lambda} \in \mathbb{R}^m, \bar{\mu} \in \mathbb{R}^p : \bar{\mu} \geq \mathbf{0}, \sum_{i=1}^{p} \mu_i = 1 \right\} \end{array} \tag{3.8}$$

This system corresponds to the set of constraints that have to be incorporated for the case of infeasible primal problems.

The dual representation of $v(\boldsymbol{y})$ is the pointwise infimum of a collection of functions that support it.

(3.9)

$$v(y) = \left[\begin{array}{rl} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}, \boldsymbol{y}) \\ \text{s.t.} & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) \;=\; \mathbf{0} \\ & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \;\leq\; \mathbf{0} \\ & \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \end{array} \right] = \left[\sup_{\lambda, \mu \geq 0} \quad \min_{\boldsymbol{x} \in \boldsymbol{X}} L(\boldsymbol{x}, \boldsymbol{y}, \lambda, \mu) \right] \quad \forall \boldsymbol{y} \in \boldsymbol{Y} \cap \boldsymbol{V}$$

$$\text{where} \quad L(\boldsymbol{x}, \boldsymbol{y}, \lambda, \mu) \;=\; f(\boldsymbol{x}, \boldsymbol{y}) + \lambda^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) + \mu^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}).$$

Now, the representation for $\boldsymbol{V}$ (3.8) and the representation for $v(\boldsymbol{y})$ (3.9) are substituted into problem 3.7 and the scalar μ_b is introduced to obtain the following master problem:

$$\begin{array}{rrcl} \min\limits_{\boldsymbol{y} \in \boldsymbol{Y}, \mu_B} & \mu_B & & \\ \text{s.t.} & \mu_B & \geq & \min\limits_{\boldsymbol{x} \in \boldsymbol{X}} L(\boldsymbol{x}, \boldsymbol{y}, \lambda, \mu) \quad \forall \lambda, \forall \mu \geq 0 \\ & 0 & \geq & \min\limits_{\boldsymbol{x} \in \boldsymbol{X}} \bar{L}(\boldsymbol{x}, \boldsymbol{y}, \bar{\lambda}, \bar{\mu}) \quad \forall \left(\bar{\lambda}, \bar{\mu}\right) \in \Lambda \end{array} \tag{3.10}$$

$$\begin{array}{rl} \text{where} & L(\boldsymbol{x}, \boldsymbol{y}, \lambda, \mu) = f(\boldsymbol{x}, \boldsymbol{y}) + \lambda^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) + \mu^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \\ & \bar{L}(\boldsymbol{x}, \boldsymbol{y}, \bar{\lambda}, \bar{\mu}) = \bar{\lambda}^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) + \bar{\mu}^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \end{array} \tag{3.11}$$

The key issue in the development of an algorithmic implementation of **GBD** is the solution of the master problem. The master problem consists of an outer optimization with respect to $\boldsymbol{y}$ whose constraints are two optimization problems with respect to $\boldsymbol{x}$ corresponding to the feasible and infeasible primal problems. These inner optimization problems need to be considered for all possible values of the Lagrange multipliers which implies that an infinite number of constraints need to be considered for the master problem.

One way to solve the master problem is to use relaxation of the problem where only a few of the constraints are considered. The inner optimization problems are considered only for fixed values of the multipliers which correspond to the multipliers from the solution of the primal problem. Furthermore, the inner optimization problems can be eliminated by evaluating the Lagrange function for fixed values of the $\boldsymbol{x}$ variables corresponding to the solution of the primal problem. This elimination assumes that the Lagrange function evaluated at the solution to the corresponding primal is a valid underestimator of the inner optimization problem. This is true when the projected problem $v(\boldsymbol{y})$ is convex in $\boldsymbol{y}$.

3.3. GBD Algorithm. The algorithm for **GBD** is stated as follows:

Step 1

Obtain initial values: $\boldsymbol{y}^1$
Set the counter: $k = 1$
Set the lower bound: $LBD = -\infty$

Set the upper bound: $UBD = +\infty$
Initialize the feasibility set: $\mathbf{F} = \emptyset$
Initialize the infeasibility set: $\bar{\mathbf{F}} = \emptyset$
Set the convergence tolerance: $\epsilon \geq 0$

Step 2

Solve the primal problem for the fixed values of $\boldsymbol{y} = \boldsymbol{y}^k$:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}, \boldsymbol{y}^k) \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^k) \leq \mathbf{0} \\ & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^k) = \mathbf{0} \end{array}$$

Obtain the optimal solution, $\boldsymbol{x}^k$, and Lagrange multipliers λ^k and μ^k
If the primal is feasible
Update the feasibility set: $\mathbf{F} = \mathbf{F} \cup k$
If the primal solution is less than the current upper bound
update the upper bound
Else
Solve the infeasible primal problem for the fixed values of $\boldsymbol{y} = \boldsymbol{y}^k$:

$$\begin{array}{lrcl} \min\limits_{\boldsymbol{x},\alpha} & \alpha_i + \alpha_e^+ + \alpha_e^- & & \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^k) - \alpha_i & \leq & \mathbf{0} \\ & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^k) + \alpha_e^+ - \alpha_e^- & = & \mathbf{0} \\ & \alpha_i, \alpha_e^+, \alpha_e^- & \geq & 0 \end{array}$$

Obtain the optimal $\bar{\boldsymbol{x}}^k$ and the Lagrange multipliers $\bar{\lambda}^k$ and $\bar{\mu}^k$
Update the infeasibility set: $\bar{\mathbf{F}} = \bar{\mathbf{F}} \cup k$

Step 3

Solve the relaxed master problem:

$$\begin{array}{llcll} \min\limits_{\boldsymbol{y},\mu_b} & \mu_b & & & \\ \text{s.t.} & \mu_b & \geq & f(\boldsymbol{x}^l, \boldsymbol{y}) + (\lambda^l)^T \boldsymbol{g}(\boldsymbol{x}^l, \boldsymbol{y}) + (\mu^l)^T \boldsymbol{h}(\boldsymbol{x}^l, \boldsymbol{y}) & l \in \mathbf{F} \\ & 0 & \geq & (\bar{\lambda}^l)^T \boldsymbol{g}(\bar{\boldsymbol{x}}^l, \boldsymbol{y}) + (\bar{\mu}^l)^T \boldsymbol{h}(\bar{\boldsymbol{x}}^l, \boldsymbol{y}) & l \in \bar{\mathbf{F}} \end{array}$$

Obtain optimal $\boldsymbol{y}^{k+1}$ and μ_b
Set the lower bound: $LBD = \mu_b$
If $UBD - LBD \leq \epsilon$
Terminate
Else
Update the counter: $k = k + 1$
Go to step 2

This algorithm can be applied to general MINLP models, however it is only guaranteed to converge to the global solution for problems which meet specific conditions. First $\boldsymbol{X}$ must be a nonempty convex set, the functions f and $\boldsymbol{g}$ must be convex for each fixed $\boldsymbol{y} \in \boldsymbol{Y}$, and the function $\boldsymbol{h}$ must be linear in $\boldsymbol{x}$ for each $\boldsymbol{y} \in \boldsymbol{Y}$.

4. Recent Developments in Global Optimization for MINLPs

Two new optimization algorithms, the Special Structure Mixed-Integer Nonlinear αBB (SMIN-αBB) and the General Structure Mixed-Integer Nonlinear αBB (GMIN-αBB), have been designed to solve large classes of nonconvex MINLPs to global optimality. They are described in the following two sections.

4.1. The SMIN-αBB Algorithm. This algorithm, proposed by [**AAF7a**], is designed to address the following class of problems to global optimality:

$$
\begin{array}{llcl}
\min & f(\boldsymbol{x}) + \boldsymbol{x}^T A_0 \boldsymbol{y} + \boldsymbol{c}_0^T \boldsymbol{y} & & \\
\text{s.t.} & \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{x}^T A_1 \boldsymbol{y} + \boldsymbol{c}_1^T \boldsymbol{y} & = & 0 \\
& \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{x}^T A_2 \boldsymbol{y} + \boldsymbol{c}_2^T \boldsymbol{y} & \leq & 0 \\
& \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{y} & \in & \{0,1\}^q
\end{array} \tag{4.1}
$$

where $\boldsymbol{c}_0^T$, $\boldsymbol{c}_1^T$ and $\boldsymbol{c}_2^T$ are constant vectors, A_0, A_1 and A_2 are constant matrices and $f(\boldsymbol{x})$, $\boldsymbol{h}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ are functions with continuous second-order derivatives.

The solution strategy for problems of type (4.1) is an extension of the αBB algorithm for twice-differentiable NLPs [**AMF95, AF96, ADFN97**]. It is based on the generation of two converging sequences of upper and lower bounds on the global optimum solution. A rigorous underestimation and convexification strategy for functions with continuous second-order derivatives allows the construction of a lower bounding MINLP problem with convex functions in the continuous variables. If no mixed-bilinear terms are present ($A_i = 0, \forall i$), the resulting MINLP can be solved to global optimality using the Outer Approximation algorithm (OA) [**DG86**]. Otherwise, the Generalized Benders Decomposition (GBD) can be used, as discussed in Section 3, or the Glover transformations [**Glo75**] can be applied to remove these bilinearities and permit the use of the OA algorithm. This convex MINLP provides a valid lower bound on the original MINLP. An upper bound on the problem can be obtained by applying the OA algorithm or the GBD to problem (4.1) to find a local solution. This bound generation strategy is incorporated within a branch-and-bound scheme: a lower and upper bound on the global solution are first obtained for the entire solution space. Subsequently, the domain is subdivided by branching on a binary or a continuous variable, thus creating new nodes for which upper and lower bounds can be computed. At each iteration, the node with the lowest lower bound is selected for branching. If the lower bounding MINLP for a node is infeasible or if its lower bound is greater than the best upper bound, this node is fathomed. The algorithm is terminated when the best lower and upper bound are within a prespecified tolerance of each other.

Before presenting the algorithmic procedure, an overview of the underestimation and convexification strategy is given, and some of the options available within the algorithm are discussed.

4.1.1. *Convex Underestimating MINLP Generation.* In order to transform an MINLP problem of the form (4.1) into a convex problem which can be solved to global optimality with the OA or GBD algorithm, the functions $f(\boldsymbol{x})$, $\boldsymbol{h}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ must be convexified. The underestimation and convexification strategy used in the αBB algorithm has previously been described in detail [**AAMF96, AF96, ADFN97**]. Its main features are exposed here.

In order to construct as tight an underestimator as possible, the nonconvex functions are decomposed into a sum of convex, bilinear, univariate concave and general nonconvex terms. The overall function underestimator can then be built by summing up the convex underestimators for all terms, according to their type. In particular, a new variable is introduced to replace each bilinear term, and is bounded by the convex envelope of the term [**AKF83**]. The univariate concave terms are linearized. For each nonconvex term $nt(\boldsymbol{x})$ with Hessian matrix $H_{nt}(\boldsymbol{x})$, a convex underestimator $\boldsymbol{L}(\boldsymbol{x})$ is defined as

$$\boldsymbol{L}(\boldsymbol{x}) = nt(\boldsymbol{x}) - \sum_i \alpha_i (x_i^U - x_i)(x_i - x_i^L) \tag{4.2}$$

where x_i^U and x_i^L are the upper and lower bounds on variable x_i respectively and the α parameters are nonnegative scalars such that $H_{nt}(\boldsymbol{x}) + 2\text{diag}(\alpha_i)$ is positive semi-definite over the domain $[\boldsymbol{x}^L, \boldsymbol{x}^U]$. The rigorous computation of the α parameters using interval Hessian matrices is described in [**AAMF96, AF96, ADFN97**].

The underestimators are updated at each node of the branch-and-bound tree as their quality strongly depends on the bounds on the variables.

4.1.2. *Branching Variable Selection.* An unusual feature of the SMIN-αBB algorithm is the strategy used to select branching variables. It follows a hybrid approach where branching may occur both on the integer and the continuous variables in order to fully exploit the structure of the problem being solved. After the node with the lowest lower bound has been identified for branching, the type of branching variable must be determined according to one of the following two criteria:

1. Branch on the binary variables first.
2. Solve a continuous relaxation of the nonconvex MINLP locally. Branch on a binary variable with a low degree of fractionality at the solution. If there is no such variable, branch on a continuous variable.

The first criterion results in the creation of an integer tree for the first q levels of the branch-and-bound tree, where q is the number of binary variables. At the lowest level of this integer tree, each node corresponds to a nonconvex NLP and the lower and upper bounding problems at subsequent levels of the tree are NLP problems. The efficiency of this strategy lies in the minimization of the number of MINLPs that need to be solved. The combinatorial nature of the problem and its nonconvexities are handled sequentially. If branching occurs on a binary variable, the selection of that variable can be done randomly or by solving a relaxation of the nonconvex MINLP and choosing the most fractional variable at the solution.

The second criterion selects a binary variable for branching only if it appears that the two newly created nodes will have significantly different lower bounds. Thus, if a variable is close to integrality at the solution of the relaxed problem, forcing it to take on a fixed value may lead to the infeasibility of one of the nodes or the generation of a high value for a lower bound, and therefore the fathoming of a branch of the tree. If no binary variable is close to integrality, a continuous variable is selected for branching.

A number of rules have been developed for the selection of a continuous branching variable. Their aim is to determine which variable is responsible for the largest separation distances between the convex underestimating functions and the original nonconvex functions. These efficient rules are exposed in [**AAF7b**].

4.1.3. *Variable Bound Updates.* Variable bound updates performed before the generation of the convex MINLP have been found to greatly enhance the speed of convergence of the αBB algorithm for continuous problems [**AAF7b**]. For continuous variables, the variable bounds are updated by minimizing or maximizing the chosen variable subject to the convexified constraints being satisfied. In spite of its computational cost, this procedure often leads to significant improvements in the quality of the underestimators and hence a noticeable reduction in the number of iterations required.

In addition to the update of continuous variable bounds, the SMIN-αBB algorithm also relies on binary variable bound updates. Through simple computations, an entire branch of the branch-and-bound tree may be eliminated when a binary variable is found to be restricted to 0 or 1. The bound update procedure for a given binary variable is as follows:

1. Set the variable to be updated to one of its bounds $y = y_B$.
2. Perform interval evaluations of the constraints in the nonconvex MINLP, using the bounds on the solution space for the current node.
3. If any of the constraints are found infeasible, fix the variable to $y = 1 - y_B$.
4. If both bounds have been tested, repeat this procedure for the next variable to be updated. Otherwise, try the second bound.

4.1.4. *Algorithmic Procedure.* The algorithmic procedure for the SMIN-αBB algorithm is formalized as follows:

Step 1

Set absolute tolerance ϵ; set $LBD^* = -\infty$ and $UBD^* = \infty$.
Set node counter $k = 0$; initialize set of nodes to be bounded, $J = \{0\}$;
$\boldsymbol{N}$, the list of nodes to be explored, is empty.

Step 2 <u>*Bounding*</u>

For each node $N_j, j \in J$:
- Perform variable bound updates if desired.
- Generate a convex lower bounding MINLP.
- Solve convex MINLP using OA or GBD. Solution is LBD_j.
- If MINLP is infeasible, fathom the current node.
- If $LBD_j \leq UBD^*$, add the current node to $\boldsymbol{N}$.
- Else, fathom the current node.

Step 3

Set LBD^* to the lowest lower bound from the list $\boldsymbol{N}$.
- If $UBD^* - LBD^* \leq \epsilon$, terminate with solution UBD^*.
- Otherwise, proceed to Step 4.

Step 4 <u>*Branching*</u>

Select the node from the list $\boldsymbol{N}$ with the lowest lower bound for branching, N_i $(i < k)$. Its lower bound is LBD_i.
Select a branching variable y^B or x^B.
Create two new regions N_{k+1} and N_{k+2}.
Set $J = k+1, k+2$ and $k = k+2$. Go back to Step 2.

4.2. The GMIN-αBB algorithm. This algorithm operates within a classical branch-and-bound framework. Branch-and-bound algorithms have been proposed by a number of researchers [**GR85, OOM90, BM91**]. The basic idea behind branch-and-bound approaches is the generation of two converging sequences of upper and lower bounds on the objective function through partitioning of the solution space. Valid lower bounds on the original MINLP are obtained by relaxing it to a continuous NLP problem, where the $\boldsymbol{y}$ variables can take on any value between 0 and 1. If the NLP relaxation has an integer solution, this solution provides an upper bound on the global solution. The generation of lower and upper bounds in this manner is referred to as the *bounding step* of the algorithm. At first, all the binary variables are relaxed and the continuous problem corresponds to the first node of a branch-and-bound tree. At the second level, two new nodes are created by forcing one of the binary variables to take on a value of 0 or 1. This is the *branching step*. Nodes in the tree are pruned when their lower bound is greater than the best upper bound on the problem, or when the relaxation is infeasible. The algorithm terminates when the lowest lower bound is within a pre-specified tolerance of the best upper bound.

The main contribution of the GMIN-αBB algorithm is its ability to identify the global optimum solution of a much larger class of problems than previously proposed algorithms. The only condition imposed on the functions $f(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y})$ in problem (2.2) is that they possess continuous second-order derivatives in the $\boldsymbol{x}$ and relaxed $\boldsymbol{y}$ space. This increased applicability is possible because of the use of the αBB global optimization algorithm for continuous twice-differentiable NLPs [**AMF95, AF96, ADFN97**]. The basic concepts behind the αBB algorithm were exposed in Section 4.1. Its use to solve the relaxed MINLP at each each node allows the identification of rigorously valid lower bounds and therefore ensures convergence to the global optimum. In general, it is not necessary to let the αBB algorithm run to completion as each one of its iterations generates a lower bound on the global solution of the NLP being solved. A strategy of early termination leads to a reduction in the computational requirements of each node of the binary branch-and-bound tree and faster overall convergence. This strategy has been used to solve a number of small nonconvex MINLP test problems as well as the pump configuration problem of [**WPG94**].

5. Algorithmic Framework

Although there are a number of algorithms available for the solution of MINLPs, there are relatively few implementations of these algorithms. The recent advances in the development of these algorithms has led to several automated implementations of these MINLP algorithms.

The first implementations made use of the modeling system **GAMS** [**BKM92**] which allows algebraic model representation and automatic interfacing with linear, nonlinear and mixed integer linear solvers. The algorithmic procedure, **APROS** [**PF89**], was developed for the automatic solution of mathematical programming problems involving decomposition techniques such as those used in the solution of MINLPs. **APROS** is an implementation of **GBD** and **OA** in **GAMS** where the modeling language is used to generate the NLP and MILP subproblems which are solved through the GAMS interface. **GAMS** also includes a direct interface to an implementation of **OA/ER** in the package **DICOPT++** [**VG90**]. The model can be written algebraically as an MINLP and the solver will perform the necessary decomposition.

More recently, the framework **MINOPT**[**SF97b**] has been developed for the solution of general mathematical programming problems. The primary motivation for its development has been brought about by the need for implementations of algorithms applicable to MINLPs. Further development has been done to address the solution of problems which involve dynamic as well as algebraic models. Extensive development of **MINOPT** has led to a highly developed computational tool.

MINOPT has a number of features including:

- Extensive implementation of optimization algorithms
- Front-end parser
- Extensive options

- Expandable platform
- Interface routines callable as a subroutine

MINOPT is capable of handling a wide variety of problems described by the variable and constraint types employed. **MINOPT** handles the following variable types:

- continuous time invariant
- continuous dynamic
- control
- integer

and recognizes the following constraint types:

- linear
- nonlinear
- dynamic
- dynamic path
- dynamic point

Different combinations of variable and constraint types lead to the following problem classifications:

- Linear Program (LP)
- Nonlinear Program (NLP)
- Mixed Integer Linear Program (MILP)
- Mixed Integer Nonlinear Program (MINLP)
- Nonlinear Program with Differential and Algebraic Constraints (NLP/DAE)
- Mixed Integer Nonlinear Program with Differential and Algebraic Constraints (MINLP/DAE)
- Optimal Control Program (OCP)
- Mixed Integer Optimal Control Program (MIOCP)

The **MINOPT** program has two phases: problem entry and problem solution. During the first phase **MINOPT** reads the input from a file, saves the problem information, and then determines the structure and consistency of the problem by analyzing the constraints and variables. After the problem has been entered, **MINOPT** proceeds to the second phase to solve the problem. Based on the problem structure determined by **MINOPT** and options supplied by the user, **MINOPT** employs the appropriate algorithm to solve the problem.

The entry phase of **MINOPT** features a parser which reads in the dynamic and/or algebraic problem formulation from an input file. The input file has a clear syntax and allows the user to enter the problem in a concise form without needing to specify the steps of the algorithm. The input file includes information such as variable names, variable partitioning (continuous, integer, dynamic), parameter definitions, and option specifications. The parser features index notation which allows for compact model representation. The parser allows for general constraint notation and has the ability to recognize and handle the various constraint types (i.e. linear, nonlinear, dynamic, point, path) and ultimately the overall structure of the problem. The **MINOPT** parser also determines the necessary analytical Jacobian information from the problem formulation.

The solution phase of **MINOPT** features extensive implementations of numerous optimization algorithms. Once the parser has determined the problem type, the solution phase applies the appropriate method to solve the problem. **MINOPT**

TABLE 1. Solution algorithms implemented by **MINOPT**

Problem Type	Algorithm	Solver
LP	Simplex method	CPLEX MINOS LSSOL
MILP	Branch and Bound	CPLEX
NLP	Augmented Lagrangian/Reduced Gradient Sequential Quadratic Programming Sequential Quadratic Programming	MINOS NPSOL SNOPT
Dynamic	Integration (Backward Difference Formula)	DASOLV
Optimal Control	Control Parameterization	DAEOPT
MINLP	Generalized Benders Decomposition Outer Approximation/Equality Relaxation Outer Approximation/Augmented Penalty Generalized Cross Decomposition	**MINOPT** **MINOPT** **MINOPT** **MINOPT**

utilizes available software packages for the solution of various subproblems. The solution algorithms implemented by **MINOPT** are listed in Table 1. The solution algorithms implemented by **MINOPT** are callable as subroutines from other programs.

MINOPT has an extensive list of options which allows the user to fine tune the various algorithms.

- selection of different algorithms for a problem type
- selection of parameters for various algorithms
- solution of the relaxed MINLP
- auto-initialization procedure—relaxed MINLP solved to determine starting values for the $\boldsymbol{y}$-variables.
- integer cuts for the **GBD** algorithm
- radial search technique for problems with discrete and continuous $\boldsymbol{y}$ variables (**GBD**)
- alternative feasibility formulation for infeasible primal
- solution of the **GBD** master problem in terms of both $\boldsymbol{x}$ and $\boldsymbol{y}$ rather than in $\boldsymbol{y}$ alone
- specification of parameters for external solvers

The flow of the program is described in Figure 3. The program is invoked from the command line and parses the input file and stores the information into a problem structure. The program then determines the appropriate method to solve the problem based on the problem type and options provided by the user. Based on the algorithm and parameters, **MINOPT** solves the problem by formulating and solving various subproblems. When needed, **MINOPT** draws necessary information from the problem structure.

The code for **MINOPT** has been written in portable ANSI C and can be compiled on any computer. **MINOPT** has been developed with an expandable platform in both the entry and solution phases of the program. This parser can be expanded to recognize additional options, variable types, commands, and constraint types that may be required of an algorithm. The solution phase of the program can be expanded to implement additional algorithms should they become available.

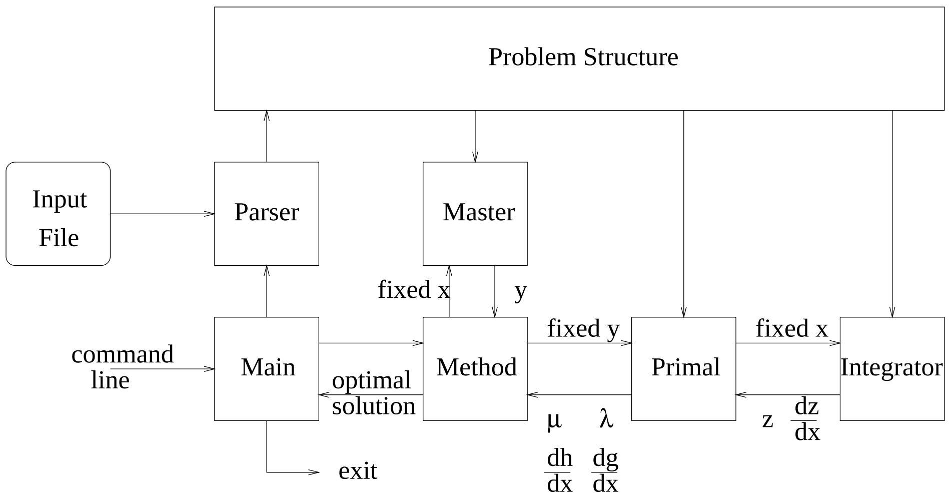

FIGURE 3. Program flow for **MINOPT**

6. Computational Study—Heat Exchanger Network Synthesis

The design of a heat exchanger network involving two hot streams, two cold streams, one hot and one cold utility is studied. The formulation of [**YG91**] is used. The annualized cost of the network is expressed as the summation of the utility costs, the fixed charges for the required heat-exchangers and an area-based cost for each each exchanger. The area is a highly nonlinear function of the heat duty and the temperature differences at both ends of the heat exchanger. The binary variables, which represent the existence of a given heat-exchanger, participate linearly in the problem. All the constraints are linear. This nonconvex MINLP therefore provides an opportunity to test the SMIN-αBB global optimization algorithm proposed in Section 4.1.

The stream data for the problem are summarized in Table 2. There are two temperature intervals. The steam utility costs \$80/kW-yr and the cooling water costs \$15/kW-yr. The fixed charges for the heat exchangers amount to \$5500/yr. The cost coefficient for the area-dependent part of the heat exchanger costs is \$300/yr. The overall heat transfer coefficients are 0.5 kW/m^{2}K for the hot stream-cold stream units, 0.83333 kW/m^{2}K for the cold stream-hot utility units and 0.5 kW/m^{2}K for the hot stream-cold utility units.

Stream	T_{in} (K)	T_{out} (K)	Fcp (kW/K)
Hot 1	650	370	10.0
Hot 2	590	370	20.0
Cold 1	410	650	15.0
Cold 2	350	500	13.0
Steam	680	680	—
Water	300	320	—

TABLE 2. Stream data for heat exchanger network problem.

The superstructure for this problem is shown in Figure 4. There are 12 possible matches and therefore 12 binary variables. The global optimum configuration involves six heat exchangers and is shown in Figure 5. Given the set ST of temperature K locations, the set HP of hot process streams and the set CP of cold process streams, the general problem formulation is as follows:

(6.1)

$$
\begin{aligned}
\min \sum_{i\in HP} & C_{CU}Q_{CU,i} + \sum_{j\in CP} C_{HU}Q_{HU,j} \\
& + \sum_{i\in HP}\sum_{j\in CP}\sum_{k\in ST} CF_{ij}z_{ijk} + \sum_{i\in HP} CF_{i,CU}z_{CU,i} + \sum_{j\in CP} CF_{j,HU}z_{HU,j} \\
& + \sum_{i\in HP}\sum_{j\in CP}\sum_{k\in ST} \frac{C_{ij}Q_{ijk}}{U_{ij}[\Delta T_{ijk}\Delta T_{ijk+1}(\Delta T_{ijk}+\Delta T_{ijk+1})/2]^{\frac{1}{3}}} \\
& + \sum_{i\in HP} \frac{C_{i,CU}Q_{CU,i}}{U_{CU,i}[\Delta T_{CU,i}(T_{out,i}-T_{in,CU})(\Delta T_{CU,i}+T_{out,i}-T_{in,CU})/2]^{\frac{1}{3}}} \\
& + \sum_{j\in CP} \frac{C_{j,HU}Q_{HU,j}}{U_{HU,j}[\Delta T_{HU,j}(T_{in,HU}-T_{out,j})(\Delta T_{HU,j}+T_{in,HU}-T_{out,j})/2]^{\frac{1}{3}}}
\end{aligned}
$$

(6.2)

$$
\begin{array}{ll}
(T_{in,i}-T_{out,i})\,Fcp_i = \sum\limits_{k\in ST}\sum\limits_{j\in CP} Q_{ijk} + Q_{CU,i} & \forall\, i\in HP \\
(T_{out,j}-T_{in,j})\,Fcp_j = \sum\limits_{k\in ST}\sum\limits_{i\in HP} Q_{ijk} + Q_{HU,j} & \forall\, j\in CP \\
(T_{i,k}-T_{i,k+1})\,Fcp_i = \sum\limits_{j\in CP} Q_{ijk} & \forall\, k\in ST,\ \forall\, i\in HP \\
(T_{j,k}-T_{j,k+1})\,Fcp_j = \sum\limits_{i\in HP} Q_{ijk} & \forall\, k\in ST,\ \forall\, j\in CP \\
T_{in,i} = T_{i,1} & \forall\, i\in HP \\
T_{in,j} = T_{j,K} & \forall\, j\in CP \\
T_{i,k} \geq T_{i,k+1} & \forall\, k\in ST,\ \forall\, i\in HP \\
T_{j,k} \geq T_{j,k+1} & \forall\, k\in ST,\ \forall\, j\in CP \\
T_{out,i} \leq T_{i,K} & \forall\, i\in HP \\
T_{out,j} \geq T_{j,1} & \forall\, j\in CP \\
(T_{i,K}-T_{out,i})\,Fcp_i = Q_{CU,i} & \forall\, i\in HP \\
(T_{out,j}-T_{j,1})\,Fcp_j = Q_{HU,j} & \forall\, j\in CP \\
Q_{ijk} - \Omega z_{ijk} \leq 0 & \forall\, k\in ST,\ \forall\, i\in HP,\ \forall\, j\in CP \\
Q_{CU,i} - \Omega z_{CU,i} \leq 0 & \forall\, i\in HP \\
Q_{HU,j} - \Omega z_{HU,j} \leq 0 & \forall\, j\in CP \\
z_{ijk}, z_{CU,i}, z_{HU,j} \in \{0,1\} & \forall\, k\in ST,\ \forall\, i\in HP,\ \forall\, j\in CP \\
T_{i,k} - T_{j,k} + \Gamma(1-z_{ijk}) \geq \Delta T_{ijk} & \forall\, k\in ST,\ \forall\, i\in HP,\ \forall\, j\in CP \\
T_{i,k+1} - T_{j,k+1} + \Gamma(1-z_{ijk}) \geq \Delta T_{ijk+1} & \forall\, k\in ST,\ \forall\, i\in HP,\ \forall\, j\in CP \\
T_{i,K} - T_{out,CU} + \Gamma(1-z_{CU,i}) \geq \Delta T_{CU,i} & \forall\, i\in HP \\
T_{out,HU} - T_{j,1} + \Gamma(1-z_{HU,j}) \geq \Delta T_{HU,j} & \forall\, j\in CP \\
\Delta T_{ijk} \geq 10 & \forall\, k\in ST,\ \forall\, i\in HP,\ \forall\, j\in CP
\end{array}
$$

where the parameters are C_{CU}, the per unit cost of cold utility; C_{HU}, the per unit cost of hot utility; CF, the fixed charged for heat exchangers; C, the area cost coefficient; T_{in}, the inlet temperature of a stream; T_{out}, the outlet temperature; Fcp, the heat capacity flowrate of a stream; Ω, the upper bound on heat exchange; Γ, the upper bound on the temperature difference. The continuous variables are T_{ik}, the temperature of hot stream i at the hot end of stage k; T_{jk}, the temperature of cold stream j at the cold end of stage k, Q_{ijk}, the heat exchanged between hot stream i and cold stream j at temperature location k; $Q_{CU,i}$, the heat exchanged between hot stream i and the cold utility at temperature location k; $Q_{HU,j}$, the heat exchanged between cold stream j and the hot utility at temperature location k; ΔT_{ijk}, the temperature approach for the match of hot stream i and cold stream j at temperature location k; $\Delta T_{CU,i}$, the temperature approach for the match of hot stream i and the cold utility at temperature location k; $\Delta T_{HU,j}$, the temperature approach for the match of cold stream j and the hot utility at temperature location k. The binary variables are z_{ijk}, for the existence of a match between hot stream i and cold stream j at temperature location k; $z_{CU,i}$, for the existence of a match between hot stream i and the cold utility at temperature location k; $z_{HU,j}$, for the existence of a match between cold stream j and the hot utility at temperature location k.

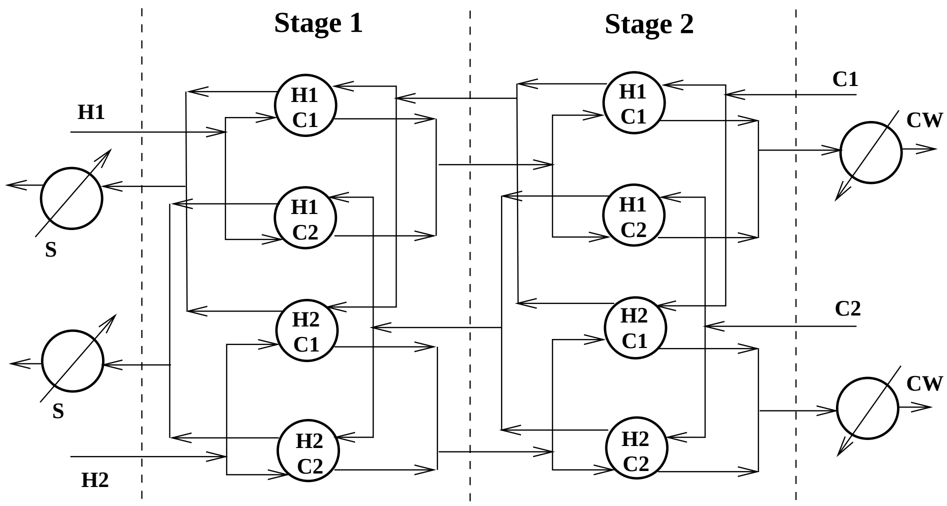

FIGURE 4. Superstructure for the heat exchanger network problem.

Due to the linear participation of the binary variables, the problem can be solved locally using the Outer Approximation algorithm [**DG86, KG87, VG90**] or the Generalized Benders Decomposition algorithm described in Section 3, and globally using the SMIN-αBB algorithm of Section 4.1.

This problem can be solved locally using **MINOPT**. For both **GBD** and **OAER** the problem is solved 30 times with random starting values for the binary variables. The starting values for the continuous variables are set to their lower bounds. The results of these runs are shown in Table 3. Whereas **GBD** generally takes more iterations than **OAER**, it converges to fewer local minima. Both algorithms obtain the global optimum roughly the same number of times.

When using the SMIN-αBB algorithm, the area-dependent cost of the heat exchangers must be underestimated using the general convex lower bounding function (4.2), in order to generate valid lower bounds on the objective function. The Outer Approximation algorithm is used to solve a lower bounding convex MINLP at each node of the tree. When this MINLP is feasible, an upper bound on the objective function is obtained by solving the nonconvex MINLP locally in the same

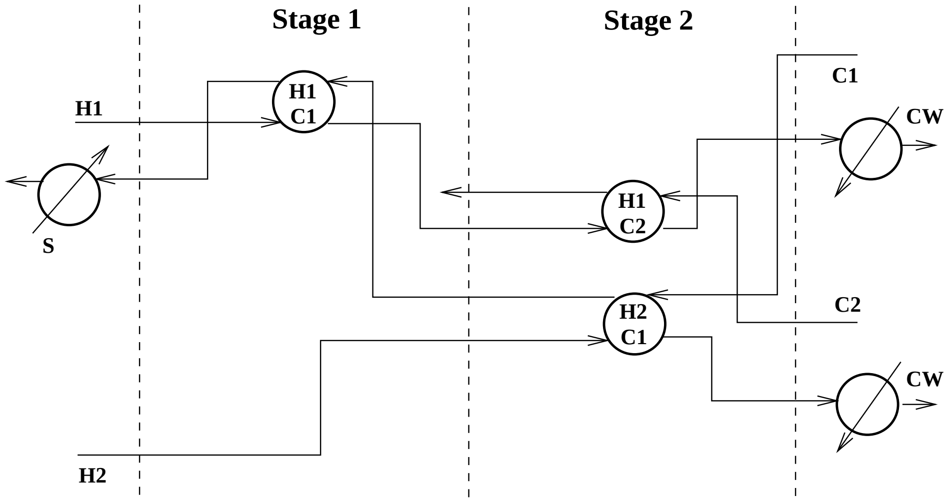

FIGURE 5. Optimum configuration for the heat exchanger network problem.

TABLE 3. Local solutions for Heat Exchanger Network Synthesis problem obtained with **MINOPT**

GBD		
Local Solutions	number of times obtained	average number of iterations
154997 (Global)	11	16
155510	18	18
161010	1	14
OAER		
Local Solutions	number of times obtained	average number of iterations
154997 (Global)	10	3.1
155510	6	3.8
167602	3	6
180848	1	5
189521	1	5
197983	7	3.6
199196	1	5
212678	1	3

region. For the heat exchanger between hot stream i and cold stream j, the convex underestimator is expressed as

$$
\begin{aligned}
&\frac{C_{ij} Q_{ijk}}{U_{ij}[\Delta T_{ijk}\Delta T_{ijk+1}(\Delta T_{ijk}+\Delta T_{ijk+1})/2]^{\frac{1}{3}}} \\
&\qquad - \alpha^{Q}_{ijk}(Q^U_{ijk} - Q_{ijk})(Q_{ijk} - Q^L_{ijk}) \\
&\qquad - \alpha^{\Delta T_k}_{ijk}(\Delta T^U_{ijk} - \Delta T_{ijk})(\Delta T_{ijk} - \Delta T^L_{ijk}) \\
&\qquad - \alpha^{\Delta T_{k+1}}_{ijk}(\Delta T^U_{ijk+1} - \Delta T_{ijk+1})(\Delta T_{ijk+1} - \Delta T^L_{ijk+1}).
\end{aligned} \tag{6.3}
$$

where α^Q_{ijk}, $\alpha^{\Delta T_k}_{ijk}$ and $\alpha^{\Delta T_{k+1}}_{ijk}$ are non-negative scalars obtained through one of the methods described by [**ADFN97**]. The convex underestimator for process stream-utility exchangers is similar, expect that one of the ΔT's is constant and only two α terms are therefore required. At the first level of the branch-and-bound tree, all binary variables can take on a value of either 0 or 1. As a result, every nonconvex term in the objective function must be underestimated to obtain a lower bound valid for the entire solution space. However, if branching occurs on the binary variables, the existence of some units is pre-determined for subsequent levels of the branch-and-bound tree. Thus, if some variable z_{ijk} is fixed to 0 at a node of the tree, proper updating of the variable bounds yields $Q_{ijk} = Q^L_{ijk} = Q^U_{ijk} = 0$. The bounds on ΔT_{ijk} and ΔT_{ijk+1} become $10 \leq \Delta T_{ijk} \leq T_{i,k} - T_{j,k} + \Gamma$ and $10 \leq \Delta T_{ijk+1} \leq T_{i,k+1} - T_{j,k+1} + \Gamma$. Since Γ is a large number, the convex terms corresponding the ΔT's do not naturally vanish from Equation (6.3). Even though the area of unit (ijk) is 0, its cost appears in the underestimating objective function as

$$
\begin{aligned}
&-\alpha^{\Delta T_k}_{ijk}(\Delta T^U_{ijk} - \Delta T_{ijk})(\Delta T_{ijk} - \Delta T^L_{ijk}) \\
&\qquad - \alpha^{\Delta T_{k+1}}_{ijk}(\Delta T^U_{ijk+1} - \Delta T_{ijk+1})(\Delta T_{ijk+1} - \Delta T^L_{ijk+1}).
\end{aligned} \tag{6.4}
$$

In order to eliminate this redundant term, it is therefore necessary to introduce modified α parameters which account for the non-existence of a unit. These new parameters are defined as

$$
\begin{aligned}
\alpha^{*,\Delta T_k}_{ijk} &= \alpha^{\Delta T_k}_{ijk}\, z^U_{ijk} \\
\alpha^{*,\Delta T_{k+1}}_{ijk} &= \alpha^{\Delta T_{k+1}}_{ijk}\, z^U_{ijk}.
\end{aligned} \tag{6.5}
$$

where z^U_{ijk} is the current upper bound on variable z_{ijk}. According to Equation (6.5), if z_{ijk} is fixed to 0, its upper bound z^U_{ijk} is 0 and $\alpha^{*,\Delta T_k}_{ijk}$ and $\alpha^{*,\Delta T_{k+1}}_{ijk}$ vanish. The convex underestimator for unit (ijk) no longer participates in the lower bounding objective function. On the contrary, if z_{ijk} is fixed to 1 or remains free to take on the value of 0 or 1, the convex underestimator is preserved.

This analysis of the objective function emphasizes the importance of the branching strategy in the generation of tight lower bounds on the objective function. Several branching strategies were used for this problem. First, the continuous variables were branched on exclusively (Run 1). Then, for Runs 2 and 3, the binary variables were branched on first, followed by the continuous variables. Finally, the "almost-integer" strategy described in Section 4.1.2 was used for Runs 4, 5 and 6. A binary variable was declared to have a low degree of fractionality if its value z^* at the solution of the relaxed MINLP was such that $\min\{z^*, 1-z^*\} \leq zdist$. For Run 4, $zdist = 0.1$ was used and for Runs 5 and 6, $zdist = 0.2$ was used.

A number of variable bound update strategies were also tested for this problem. In Runs 1 and 2, updates were performed only for the continuous variables. In all other runs, the bounds on the binary variables were also updated. In Run 6, the effect of updating the bounds on only a fraction of the continuous variables was studied.

The results are shown in Figure 6 and Table 4. Branching on the continuous variables only results in slow asymptotic convergence of algorithm to the global optimum solution (Run 1). The rate of convergence is greatly improved when the binary variables can be used for branching (Runs 2 to 6). Although the "almost-integer" branching strategy exhibits the best performance in terms of iterations (Runs 4 to 6), the lowest CPU requirements correspond to Run 3, which branches on all the binary variables before turning to the continuous variables. The average time spent on each iteration of the algorithm is therefore greater when the "almost-integer" strategy is applied. Two factors can account for this increase in the computational requirements. First, the selection of a binary branching variable requires the solution of a nonconvex MINLP. In addition, the generation of a lower bound on the solution at almost every node of the branch-and-bound tree for Runs 4 to 6 necessitates the solution of a convex MINLP. By comparison, only 58% of the nodes in the branch-and-bound tree for Run 3 involve the solution of a convex MINLP. Lower bounds at the remaining nodes are obtained by solving less expensive convex NLPs. Addressing the combinatorial aspects of the problem first by branching on the binary variables thus leads to the better performance of the SMIN-αBB algorithm.

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 4. Optimization of a heat exchanger network – Note that Run 1 converges asymptotically.

7. Conclusions

As was demonstrated in this paper, mathematical programming techniques are a valuable tool for the solution of network applications. The optimization approach to process synthesis illustrates their use for an important industrial application. It was shown that this procedure generates Mixed-Integer Nonlinear Programming problems (MINLPs) and a decomposition based method, **GBD**, capable of addressing such problems was presented. Considerable progress has been made in handling both the combinatorial aspects of the problem as well as nonconvexity issues so that the global solution of increasingly complex problems can be identified. The development of the SMIN-αBB and GMIN-αBB algorithms has extended the class of problems that can rigorously be solved to global optimality.

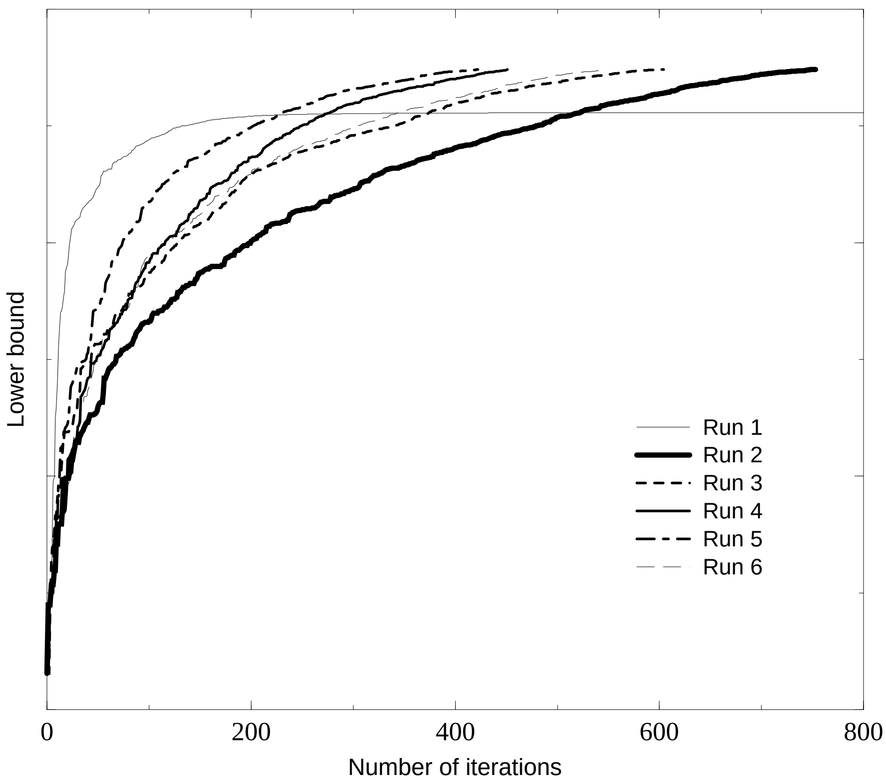

FIGURE 6. Progress of the lower bound for the heat exchanger network

The increasing capability of MINLP algorithms has permitted the development of automated frameworks such as MINOPT, in which general mathematical representations can be addressed. These developments have led researchers in numerous fields to employ mathematical modeling and numerical solution through MINLP optimization techniques in order to address their problems.

A number of issues must be resolved in order to develop algorithms that can handle more complex and realistic problems. Although computational power has increased, the ability of MINLP algorithms to solve large scale problems is still limited: a large number of integer variables leads to combinatorial problems, and a large number of continuous variables leads to the generation of large scale NLPs. In addition, rigorous models capable of accurately describing industrial operations usually involve complex mathematical expressions and result in problems which are difficult to solve using standard procedures. Finally, important challenges such as the inclusion of dynamic models and optimal control problems into the MINLP framework have only recently been addressed [**SF97a**].

References

[AAF7a] C. S. Adjiman, I. P. Androulakis, and C. A. Floudas, *Global optimization of MINLP problems in process synthesis and design*, Comput. Chem. Eng. **21** (1997a), S445–S450.

[AAF7b] C. S. Adjiman, I. P. Androulakis, and C. A. Floudas, *A global optimization method, αBB, for general twice–differentiable NLPs – II. Implementation and computational results*, accepted for publication, 1997b.

[AAMF96] C. S. Adjiman, I. P. Androulakis, C. D. Maranas, and C. A. Floudas, *A global optimisation method, αBB, for process design*, Comput. Chem. Eng. Suppl. **20** (1996), S419–S424.

[ADFN97] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier, *A global optimization method, αBB, for general twice–differentiable NLPs – I. Theoretical advances*, accepted for publication, 1997.

[AF96] C. S. Adjiman and C. A. Floudas, *Rigorous convex underestimators for general twice–differentiable problems*, J. Glob. Opt. **9** (1996), 23–40.

[AKF83] F. A. Al-Khayyal and J. E. Falk, *Jointly constrained biconvex programming*, Math. of Oper. Res. **8** (1983), 273–286.

[AMF95] I. P. Androulakis, C. D. Maranas, and C. A. Floudas, *αBB : A global optimization method for general constrained nonconvex problems*, J. Glob. Opt. **7** (1995), 337–363.

[Bea77] E. M. L. Beale, The State of the Art in Numerical Analysis, ch. Integer programming, pp. 409–448, Academic Press, 1977, pp. 409–448.

[Ben62] J. F. Benders, *Partitioning procedures for solving mixed-variables programming problems*, Numer. Math. **4** (1962), 238.

[BKM92] Anthony Brooke, David Kendrick, and Alexander Meeraus, *Gams: A user's guide*, Boyd & Fraser, Danvers, MA, 1992.

[BM91] B. Borchers and J. E. Mitchell, *An improved branch and bound algorithm for mixed integer nonlinear programs*, Tech. Report RPI Math Report No. 200, Renssellaer Polytechnic Institute, 1991.

[DG86] M. A. Duran and I. E. Grossmann, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Math. Prog. **36** (1986), 307–339.

[FAC89] C. A. Floudas, A. Aggarwal, and A. R. Ciric, *Global optimal search for nonconvex NLP and MINLP problems*, Comput. Chem. Eng. **13** (1989), no. 10, 1117.

[FL94] R. Fletcher and S. Leyffer, *Solving mixed integer nonlinear programs by outer approximation*, Math. Prog. **66** (1994), no. 3, 327.

[Flo95] C. A. Floudas, *Nonlinear and mixed integer optimization: Fundamentals and applications*, Oxford University Press, 1995.

[Geo72] A. M. Geoffrion, *Generalized benders decomposition*, J. Opt. Theory Applic. **10** (1972), no. 4, 237–260.

[Glo75] F. Glover, *Improved linear integer programming formulations of nonlinear integer problems*, Management Sci. **22** (1975), no. 4, 445.

[GR85] O. K. Gupta and R. Ravindran, *Branch and bound experiments in convex nonlinear integer programing*, Management Sci. **31** (1985), no. 12, 1533–1546.

[Gup80] O. K. Gupta, *Branch and bound experiments in nonlinear integer programming*, Ph.D. thesis, Purdue University, 1980.

[Hol90] K. Holmberg, *On the convergence of the cross decomposition*, Math. Prog. **47** (1990), 269.

[KG87] G. R. Kocis and I. E. Grossmann, *Relaxation strategy for the structural optimization of process flow sheets*, Ind. Eng. Chem. Res. **26** (1987), no. 9, 1869.

[KG89] G. R. Kocis and I. E. Grossmann, *A modelling and decomposition strategy for the MINLP optimization of process flowsheets*, Comput. Chem. Eng. **13** (1989), no. 7, 797–819.

[MM86] H. Mawengkang and B. A. Murtagh, *Solving nonlinear integer programs with large scale optimization software*, Ann. of Oper. Res. **5** (1986), 425.

[OOM90] G. M. Ostrovsky, M. G. Ostrovsky, and G. W. Mikhailow, *Discrete optimization of chemical processes*, Comput. Chem. Eng. **14** (1990), no. 1, 111.

[PF89] G. E. Paules, IV and C. A. Floudas, *APROS: Algorithmic development methodology for discrete-continuous optimization problems*, Oper. Res. **37** (1989), no. 6, 902–915.

[QG92] I. Quesada and I. E. Grossmann, *An LP/NLP based branch and bound algorithm for convex MINLP optimization problems*, Comput. Chem. Eng. **16** (1992), no. 10/11, 937–947.

[RG94] R. Raman and I. E. Grossmann, *Modeling and computational techniques for logic based integer programming*, Comput. Chem. Eng. **18** (1994), 563–578.

[SF97a] C. A. Schweiger and C. A. Floudas, *Interaction of design and control: Optimization with dynamic models*, Optimal Control: Theory, Algorithms, and Applications (W. W. Hager and P. M. Pardalos, eds.), Kluwer Academic Publishers, 1997, accepted for publication.

[SF97b] C. A. Schweiger and C. A. Floudas, *MINOPT: A software package for mixed-integer nonlinear optimization*, Princeton University, Princeton, NJ 08544-5263, 1997, Version 2.0.

[TG96] M. Türkay and I. E. Grossmann, *Logic-based MINLP algorithms for the optimal synthesis of process networks*, Comput. Chem. Eng. **20** (1996), no. 8, 959–978.

[VEH96] R. Vaidyanathan and M. El-Halwagi, *Global optimization of nonconvex MINLP's by interval analysis*, Global Optimization in Engineering Design (I. E. Grossmann, ed.), Kluwer Academic Publishers, 1996, pp. 175–193.

[VG90] J. Viswanathan and I. E. Grossmann, *A combined penalty function and outer approximation method for MINLP optimization*, Comput. Chem. Eng. **14** (1990), no. 7, 769–782.

[WP95] T. Westerlund and F. Pettersson, *An extended cutting plane method for solving convex MINLP problems*, Comput. Chem. Eng. Suppl. **19** (1995), 131–136.

[WPG94] T. Westerlund, F. Pettersson, and I. E. Grossmann, *Optimization of pump configuration problems as a MINLP probem*, Comput. Chem. Eng. **18** (1994), no. 9, 845–858.

[YG91] T. F. Yee and I. E. Grossmann, *Simultaneous optimization model for heat exchanger network synthesis*, Chemical Engineering Optimization Models with GAMS (I. E. Grossmann, ed.), CACHE Design Case Studies Series, vol. 6, 1991.

Department of Chemical Engineering, Princeton University, Princeton, New Jersey 08544-5263

E-mail address: `claire@titan.princeton.edu`

Department of Chemical Engineering, Princeton University, Princeton, New Jersey 08544-5263

E-mail address: `carl@titan.princeton.edu`

Department of Chemical Engineering, Princeton University, Princeton, New Jersey 08544-5263

E-mail address: `floudas@titan.princeton.edu`