HANDBOOK OF COMBINATORIAL OPTIMIZATION
D.-Z. Du and P.M. Pardalos (Eds.) pp. – - –

Mixed-Integer Nonlinear Optimization in Process Synthesis

C. S. Adjiman, C. A. Schweiger, and C. A. Floudas†
Department of Chemical Engineering
Princeton University, Princeton, NJ 08544-5263
E-mail: `claire@titan.princeton.edu`, `carl@titan.princeton.edu`, `floudas@titan.princeton.edu`

†Author to whom all correspondence should be addressed.

Contents

1 Introduction 3

2 Optimization Approach in Process Synthesis 5

3 Algorithms for Convex MINLPs 8
3.1 Generalized Benders Decomposition . . . 9
3.1.1 Primal Problem . . . 10
3.1.2 Master Problem . . . 11
3.1.3 **GBD** Algorithm . . . 13
3.2 Outer Approximation . . . 14
3.2.1 Outer Approximation Primal Problem . . . 15
3.2.2 Outer Approximation Master Problem . . . 15
3.2.3 **OA** Algorithm . . . 16
3.3 Outer Approximation/Equality Relaxation . . . 18
3.3.1 **OA/ER** Algorithm . . . 18
3.4 Outer Approximation/Equality Relaxation/Augmented Penalty . . . 20
3.4.1 **OA/ER/AP** Algorithm . . . 21
3.5 Generalized Outer Approximation . . . 23
3.5.1 Primal Problem . . . 23
3.5.2 Master Problem . . . 23
3.5.3 **GOA** Algorithm . . . 25
3.6 Generalized Cross Decomposition . . . 26

3.6.1 **GCD** Algorithm 27
3.7 Branch-and-Bound Algorithms 30
3.7.1 Selection of the Branching Node 32
3.7.2 Selection of the Branching Variable 32
3.7.3 Generation of a Lower Bound 33
3.7.4 Algorithmic Procedure 33
3.8 Extended Cutting Plane (ECP) 34
3.8.1 Algorithmic Procedure 36
3.9 Feasibility Approach 36
3.10 Logic Based Approach 37
3.10.1 Logic-Based Outer Approximation 38
3.10.2 Logic-Based Generalized Benders Decomposition 40

4 Global Optimization for Nonconvex MINLPs **40**
4.1 Branch-and-reduce algorithm 41
4.2 Interval Analysis Based Algorithm 42
4.2.1 Node Fathoming Tests for Interval Algorithm 42
4.2.2 Branching Step 43
4.3 Extended Cutting Plane for Nonconvex MINLPs 44
4.4 Reformulation/Spatial Branch-and-Bound Algorithm 45
4.5 The SMIN-αBB Algorithm 46
4.5.1 Convex Underestimating MINLP Generation 47
4.5.2 Branching Variable Selection 47
4.5.3 Variable Bound Updates 48
4.5.4 Algorithmic Procedure 49
4.6 The GMIN-αBB algorithm 50

5 Implementation: MINOPT **51**

6 Computational Studies **56**
6.1 Distillation Sequencing 56
6.2 Heat Exchanger Network Synthesis 60
6.2.1 Solution Strategy with the SMIN-αBB algorithm 66
6.2.2 Specialized Algorithm for Heat Exchanger Network Problems 70

7 Conclusions **70**

8 Acknowledgments **71**

References

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, developed for the solution of this class of problems, are presented in this paper: Generalized Benders Decomposition (GBD), Outer Approximation (OA), Generalized Cross Decomposition (GCD), Branch and Bound (BB), Extended Cutting Plane (ECP), and Feasibility Approach (FA). Some 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 the theoretical issues behind local and global optimization algorithms for MINLPs have been exposed, attention is directed to their algorithmic development and implementation. The 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 use of **MINOPT** is illustrated through the solution of a variety of process network problems. The synthesis problem for a heat exchanger network is then presented to demonstrate the global optimization SMIN-αBB algorithm.

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, as well as total process flowsheets.

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 optimization algorithms developed for the solution of the posed optimization problem are presented. Although these methods 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 the described algorithms. The final part of the paper describes the application of both global and local methods to a heat exchanger network synthesis problem.

2 Optimization Approach in Process Synthesis

A major advance in process synthesis has been the development of the optimization approach to the process synthesis problem. 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 be 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.

Another example of a superstructure is illustrated by the two compo-

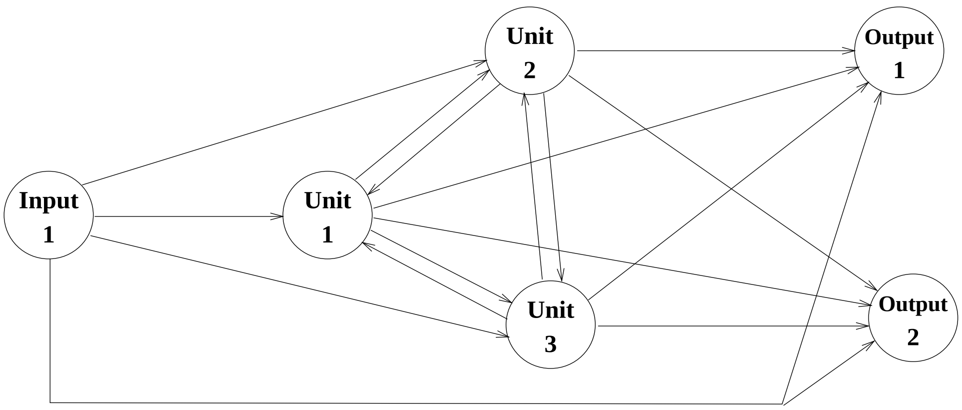

Figure 1: Network representation of superstructure

nent 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.

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):

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

where

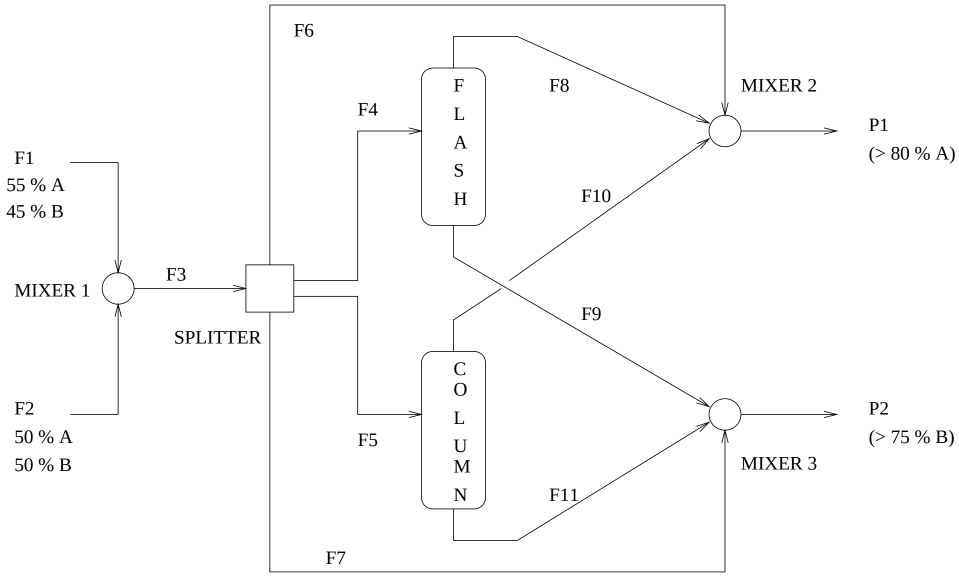

Figure 2: A Two-Column Distillation Sequence Superstructure

- $\boldsymbol{x}$ 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

$$
\begin{array}{lrcl}
\min\limits_{\boldsymbol{x},\boldsymbol{y}} & f(\boldsymbol{x},\boldsymbol{y}) & & \\
\text{s.t.} & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) & = & \mathbf{0} \\
 & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) & \leq & \mathbf{0} \\
 & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
 & \boldsymbol{y} & \in & \{0,1\}^q
\end{array}
\tag{2}
$$

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 review of these developments is presented in the next section.

3 Algorithms for Convex MINLPs

A number of algorithms have been developed to address problems with the above form 2. Some deal with the formulation as stated, while others deal with a restricted class of the problem. The following is a chronological listing of these algorithms.

1. Generalized Benders Decomposition, **GBD** [Geo72, PF89, FAC89]

2. Branch and Bound, **BB** [Bea77, Gup80, OOM90, BM91]

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];

An overview of these MINLP algorithms and extensive theoretical, algorithmic, and applications-oriented descriptions 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.

3.1 Generalized Benders Decomposition

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. 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 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.

3.1.1 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:

$$\begin{array}{lrcl} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}, \boldsymbol{y}^k) & & \\ \text{s.t.} & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^k) & = & \mathbf{0} \\ & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^k) & \leq & \mathbf{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \end{array} \tag{3}$$

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:

$$\begin{array}{lrcl} \min\limits_{\boldsymbol{x}, \alpha} & \alpha_i + \alpha_e^+ + \alpha_e^- & & \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^k) - \alpha_i & \leq & \mathbf{0} \\ & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^k) + \alpha_e^+ - \alpha_e^- & = & \mathbf{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \alpha_i, \alpha_e^+, \alpha_e^- & \geq & 0 \end{array} \tag{4}$$

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

$$
\begin{array}{ll}
\min\limits_{\boldsymbol{x},\alpha} & \alpha \\
\text{s.t.} & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}^k) \leq \alpha \\
& \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}^k) = \mathbf{0} \\
& \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \alpha \geq 0
\end{array} \tag{5}
$$

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.

3.1.2 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 can be written as

$$
\begin{array}{lllll}
\min\limits_{\boldsymbol{y}} & \inf\limits_{\boldsymbol{x}} & f(\boldsymbol{x},\boldsymbol{y}) & & \\
& \text{s.t.} & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) & = & \mathbf{0} \\
& & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) & \leq & \mathbf{0} \\
& & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
& & \boldsymbol{y} & \in & \boldsymbol{Y} \in \{0,1\}^q
\end{array} \tag{6}
$$

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

$$
\begin{array}{lllll}
v(\boldsymbol{y}) = & \inf\limits_{\boldsymbol{x}} & f(\boldsymbol{x},\boldsymbol{y}) & & \\
& \text{s.t.} & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) & = & \mathbf{0} \\
& & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) & \leq & \mathbf{0} \\
& & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n
\end{array} \tag{7}
$$

$$
\boldsymbol{V} = \{\boldsymbol{y} : \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = \mathbf{0}, \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq \mathbf{0} \quad \text{for some} \quad \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n\} \tag{8}
$$

The projected problem can now be written as:

$$
\begin{array}{ll}
\min\limits_{\boldsymbol{y}} & v(\boldsymbol{y}) \\
\text{s.t.} & \boldsymbol{y} \in \boldsymbol{Y} \cap \boldsymbol{V}
\end{array} \tag{9}
$$

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:

$$\begin{array}{rcl} \mathbf{0} & \geq & \inf \bar{L}(\boldsymbol{x}, \boldsymbol{y}, \bar{\lambda}, \bar{\mu}), \quad \forall \bar{\lambda}, \bar{\mu} \in \Lambda \\ \text{where} \quad \Lambda & = & \left\{ \bar{\lambda} \in \mathbb{R}^m, \bar{\mu} \in \mathbb{R}^p : \bar{\mu} \geq \mathbf{0}, \sum_{i=1}^{p} \mu_i = 1 \right\} \end{array} \tag{10}$$

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.

$$\begin{array}{rl} v(y) & = \left[\begin{array}{rrcl} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}, \boldsymbol{y}) & & \\ \text{s.t.} & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) & = & \mathbf{0} \\ & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) & \leq & \mathbf{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \end{array} \right] \\ & = \left[\sup\limits_{\lambda, \mu \geq 0} \quad \min\limits_{\boldsymbol{x} \in \boldsymbol{X}} L(\boldsymbol{x}, \boldsymbol{y}, \lambda, \mu) \right] \quad \forall \boldsymbol{y} \in \boldsymbol{Y} \cap \boldsymbol{V} \end{array} \tag{11}$$

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

Now, the representation for $\boldsymbol{V}$ (10) and the representation for $v(\boldsymbol{y})$ (11 are substituted into problem 9 and the scalar μ_b is introduced to obtain the following master problem:

$$\begin{array}{rrcl} \min\limits_{\boldsymbol{y} \in \boldsymbol{Y}, \mu_B} & \mu_B & & \\ \text{s.t.} & \mu_B & \geq & \min\limits_{\boldsymbol{x} \in \boldsymbol{X}} L(\boldsymbol{x}, \boldsymbol{y}, \lambda, \mu) \quad \forall \lambda, \forall \mu \geq 0 \\ & 0 & \geq & \min\limits_{\boldsymbol{x} \in \boldsymbol{X}} \bar{L}(\boldsymbol{x}, \boldsymbol{y}, \bar{\lambda}, \bar{\mu}) \quad \forall \left(\bar{\lambda}, \bar{\mu}\right) \in \Lambda \end{array} \tag{12}$$

$$\begin{array}{rl} \text{where} & L(\boldsymbol{x}, \boldsymbol{y}, \lambda, \mu) = f(\boldsymbol{x}, \boldsymbol{y}) + \lambda^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) + \mu^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \\ & \bar{L}(\boldsymbol{x}, \boldsymbol{y}, \bar{\lambda}, \bar{\mu}) = \bar{\lambda}^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) + \bar{\mu}^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \end{array} \tag{13}$$

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 $\boldsymbol{y}$.

3.1.3 GBD Algorithm

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 $\boldsymbol{y} = \boldsymbol{y}^k$:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}, \boldsymbol{y}^k) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^k) \leq \mathbf{0} \\ & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^k) = \mathbf{0} \end{aligned}$$

Obtain the optimal solution, optimal $\boldsymbol{x}^k$ and optimal Lagrange multipliers λ^k and μ^k
If the primal is feasible
Update the feasibility set: $\mathbf{F} = \mathbf{F} \cup k$
If the optimal solution of the primal is less than UBD update the upper bound (UBD)
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 $\boldsymbol{y} \in \boldsymbol{Y}$.

3.2 Outer Approximation

The basic ideas behind the Outer Approximation methods [DG86] is similar to those for **GBD**. At each iteration, upper and lower bounds on the solution to the MINLP are generated. The upper bound results from the solution of a primal problem which is formulated identically to the primal problem for **GBD**. The lower bound is determined by solving a master problem which is an outer linearization of the problem around the primal solution.

The outer approximation methods deal with a subclass of MINLP problems in which the functions $f(\boldsymbol{x}, \boldsymbol{y})$, and $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})$ are linear in the $\boldsymbol{y}$ variables and separable in $\boldsymbol{x}$ and $\boldsymbol{y}$. The set of $\boldsymbol{y}$ variables is also strictly binary variables. The formulation also does not allow for equality constraints. Thus, any equality constraints must be eliminated either algebraically or numerically in order to apply the **OA** algorithm. This class of MINLPs has the following formulation:

$$\begin{array}{lrcl} \min\limits_{\boldsymbol{x},\boldsymbol{y}} & f(\boldsymbol{x}) + \boldsymbol{c}^T\boldsymbol{y} & & \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{c}^T\boldsymbol{y} & \leq & \mathbf{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} & \in & \{0,1\}^q \end{array} \tag{14}$$

The **OA** algorithm is similar to the **GBD** algorithm in that it iterates between upper and lower bounding primal and master subproblems. The difference is in the formulation of the master problem. The master problem for this method is formed by a projection onto the $\boldsymbol{y}$ space and an outer approximation of the objective function and feasible region.

3.2.1 Outer Approximation Primal Problem

As in **GBD**, the primal problem results from fixing the values of the $\boldsymbol{y}$ variables to $\boldsymbol{y}^k$:

$$\begin{array}{lrcl} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) + \boldsymbol{c}^T\boldsymbol{y}^k & & \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{y}^k & \leq & \mathbf{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \end{array} \tag{15}$$

If this problem is feasible, its solution provides an upper bound on the solution of the MINLP model. If the primal is infeasible, a feasibility problem similar to those used in **GBD** is formulated and solved.

3.2.2 Outer Approximation Master Problem

The master problem is formulated by projecting the problem onto the $\boldsymbol{y}$ space and using an outer approximation of the objective function and feasible region. The projected problem can be written as

$$\begin{array}{ll} \min\limits_{\boldsymbol{y}} & v(\boldsymbol{y}) \\ \text{s.t.} & \boldsymbol{y} \in \boldsymbol{Y} \cap \boldsymbol{V} \end{array} \tag{16}$$

where

$$v(\boldsymbol{y}) = \boldsymbol{c}^T\boldsymbol{y} + \begin{array}{ll} \inf\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{B}^T\boldsymbol{y} \leq \mathbf{0} \\ & \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \end{array} \tag{17}$$

and

$$\boldsymbol{V} = \{\boldsymbol{y} : \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{y} \le 0, \text{ for some } \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n\} \tag{18}$$

The outer approximation of $v(\boldsymbol{y})$ is performed by linearizing $f(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ around the solution of the primal problem $\boldsymbol{x}^k$. Provided that the functions $f(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ are convex, the linearizations represent valid support functions. Thus, replacing $v(\boldsymbol{y})$ with its outer approximation and replacing $\boldsymbol{y} \in \boldsymbol{Y} \cap \boldsymbol{V}$ with $\boldsymbol{y} \in \boldsymbol{V}$ along with an integer cut constraint, the following master problem results:

$$\begin{array}{ll}
\min\limits_{\boldsymbol{x},\boldsymbol{y},\mu_{\text{OA}}} & \boldsymbol{c}^T\boldsymbol{y} + \mu_{\text{OA}} \\
\text{s.t.} & \left.\begin{array}{rcl} \mu_{\text{OA}} & \ge & f(\boldsymbol{x}^k) + \nabla_x f(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) \\ \mathbf{0} & \ge & \boldsymbol{g}(\boldsymbol{x}^k) + \nabla_x \boldsymbol{g}(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) + \boldsymbol{B}\boldsymbol{y} \end{array}\right\} \forall k \in \mathbf{F} \\
& \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{y} \in \boldsymbol{Y} \in \{0,1\} \\
& \sum\limits_{i \in \mathbf{B}^k} y_i^k - \sum\limits_{i \in \mathbf{NB}^k} y_i^k \le \left|\mathbf{B}^k\right| - 1, \quad k \in \mathbf{F}
\end{array} \tag{19}$$

where $\boldsymbol{F}$ is the set of all feasible solutions $\boldsymbol{x}^k$ to the primal problem.

Since the $\boldsymbol{y}$ variables participate linearly and are binary variables, this formulation is an MILP which can be solved by standard branch-and-bound algorithms. This formulation of the master problem requires that all of the feasible solutions to the primal problem be known which implies an exhaustive enumeration of the binary variables. In order to accommodate for this inefficiency, a relaxation is proposed where only linearizations around the currently known feasible points are included in the master problem. Additionally, to ensure that integer combinations which produce infeasible primal problems are also infeasible in the master problem, linearizations about the solution to the feasibility problem are also included in the master problem.

3.2.3 OA Algorithm

Step 1

Obtain starting point: $\boldsymbol{y}^1$
Set the counter: $k = 1$
Set the lower bound: $LBD = -\infty$
Set the upper bound: $UBD = +\infty$
Set the convergence tolerance: $\epsilon \ge 0$

Step 2

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

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) + \boldsymbol{c}^T \boldsymbol{y}^k \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq -\boldsymbol{B}\boldsymbol{y}^k \end{aligned}$$

Obtain the optimal $\boldsymbol{x}^k$

If the primal is feasible

If the optimal solution of the primal is less than UBD

Update the upper bound (UBD)

Else

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

$$\begin{aligned} \min_{\boldsymbol{x},\alpha} \quad & \alpha_i + \alpha_e^+ + \alpha_e^- \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) - \alpha_i \leq -\boldsymbol{B}\boldsymbol{y}^k \\ & \alpha_i, \alpha_e^+, \alpha_e^- \geq 0 \end{aligned}$$

Obtain the optimal $\bar{\boldsymbol{x}}^k$

Step 3

Solve the relaxed master problem:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y},\mu_{\mathrm{OA}}} \quad & \boldsymbol{c}^T \boldsymbol{y} + \mu_{\mathrm{OA}} \\ \text{s.t.} \quad & \left.\begin{aligned} \mu_{\mathrm{OA}} &\geq f(\boldsymbol{x}^l) + \nabla_x f(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) \\ \boldsymbol{0} &\geq \boldsymbol{g}(\boldsymbol{x}^l) + \nabla_x \boldsymbol{g}(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) + \boldsymbol{B}\boldsymbol{y} \end{aligned}\right\} \forall l \in \mathbf{F} \\ & \boldsymbol{0} \geq \boldsymbol{g}(\bar{\boldsymbol{x}}^l) + \nabla_x \boldsymbol{g}(\bar{\boldsymbol{x}}^l)(\boldsymbol{x} - \bar{\boldsymbol{x}}^l) + \boldsymbol{B}\boldsymbol{y} \;\Big\}\; \forall l \in \bar{\mathbf{F}} \\ & \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} \in \boldsymbol{Y} \in \{0,1\} \\ & \sum_{i \in \mathbf{B}^l} y_i^l - \sum_{i \in \mathbf{NB}^l} y_i^l \leq \left|\mathbf{B}^l\right| - 1, \quad \forall l \in \mathbf{F} \end{aligned}$$

Obtain the solution and $\boldsymbol{y}^{k+1}$

If the solution to the master is greater than the current lower bound

Update the lower bound.

If $UBD - LBD \leq \epsilon$

Terminate

Else

Update the counter: $k = k + 1$

Go to step 2

The stated algorithm may be applied to general problems whose formulation is of the form (14). However, it is not guaranteed to converge to the global solution unless some additional conditions are met. The functions f and $\boldsymbol{g}$ must be convex in $\boldsymbol{x}$. If this is not the case, the linearizations employed in the master problem may not be valid and may possibly eliminate part of the feasible region.

3.3 Outer Approximation/Equality Relaxation

The **OA/ER** algorithm is a generalization of the Outer Approximation algorithm [KG87] to handle nonlinear equality constraints. The class of problems this algorithm can address is the following:

$$\begin{array}{lrcl} \min\limits_{\boldsymbol{x},\boldsymbol{y}} & f(\boldsymbol{x}) + \boldsymbol{c}^T\boldsymbol{y} & & \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{y} & \leq & \mathbf{0} \\ & \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{C}\boldsymbol{y} & = & \mathbf{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} & \in & \{0,1\}^q \end{array} \tag{20}$$

The basic idea behind this algorithm is to relax the equality constraints into inequalities and apply the **OA** algorithm. A square diagonal matrix $\boldsymbol{T}$, whose diagonal elements are minus one, zero and one, is used for relaxing the equality constraints.

$$\boldsymbol{T}^k\left(\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{C}\boldsymbol{y}\right) \leq \mathbf{0}$$

The matrix T has the same number of rows as the number of equality constraints and the values of the diagonal elements depend on the signs of the corresponding multipliers obtained from the primal problem. The values of the elements are one for the multipliers which are positive, minus one for the negative multipliers, and zero for the zero valued multipliers.

$$\boldsymbol{T}^k = \text{diag}(t_{ii}) \quad t_{ii} = \text{sign}(\mu_i^k)$$

With the equalities now relaxed to inequalities, the principles of the **OA** can be applied to the problem.

3.3.1 OA/ER Algorithm

Step 1

Obtain starting point: $\boldsymbol{y}^1$
Set the counter: $k = 1$
Set the lower bound: $LBD = -\infty$
Set the upper bound: $UBD = +\infty$
Set the convergence tolerance: $\epsilon \geq 0$

Step 2

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

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) + \boldsymbol{c}^T \boldsymbol{y}^k \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq -\boldsymbol{B}\boldsymbol{y}^k \\ & \boldsymbol{h}(\boldsymbol{x}) = -\boldsymbol{C}\boldsymbol{y}^k \end{aligned}$$

Obtain the optimal $\boldsymbol{x}^k$ and the Lagrange multipliers μ^k

If the primal is feasible

If the optimal solution of the primal is less than UBD

Update the upper bound (UBD)

Else

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

$$\begin{array}{lrcl} \min\limits_{x,\alpha} & \multicolumn{3}{l}{\alpha_i + \alpha_e^+ + \alpha_e^-} \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) - \alpha_i & \leq & -\boldsymbol{B}\boldsymbol{y}^k \\ & \boldsymbol{h}(\boldsymbol{x}) + \alpha_e^+ - \alpha_e^- & = & -\boldsymbol{C}\boldsymbol{y}^k \\ & \alpha_i, \alpha_e^+, \alpha_e^- & \geq & 0 \end{array}$$

Obtain the optimal $\bar{\boldsymbol{x}}^k$ and Lagrange multipliers $\bar{\mu}^k$

Step 3

Determine the matrix $\boldsymbol{T}$:

$\boldsymbol{T}^k = \mathrm{diag}(t_{ii})$ where $t_{ii} = \mathrm{sign}(\mu_i^k)$

Solve the relaxed master problem:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x},\boldsymbol{y}} & \mu + \boldsymbol{c}^T \boldsymbol{y} \\ \text{s.t.} & \left.\begin{array}{rcl} \mu & \geq & f(\boldsymbol{x}^l) + \nabla_x f(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) \\ \mathbf{0} & \geq & \boldsymbol{g}(\boldsymbol{x}^l) + \nabla_x \boldsymbol{g}(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) + \boldsymbol{B}\boldsymbol{y} \\ \mathbf{0} & \geq & \boldsymbol{T}^l(\boldsymbol{h}(\boldsymbol{x}^l) + \nabla_x \boldsymbol{h}(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) + \boldsymbol{C}\boldsymbol{y}) \end{array}\right\} \forall l \in \mathbf{F} \\ & \left.\begin{array}{rcl} \mathbf{0} & \geq & \boldsymbol{g}(\bar{\boldsymbol{x}}^l) + \nabla_x \boldsymbol{g}(\bar{\boldsymbol{x}}^l)(\boldsymbol{x} - \bar{\boldsymbol{x}}^l) + \boldsymbol{B}\boldsymbol{y} \\ \mathbf{0} & \geq & \boldsymbol{T}^l(\boldsymbol{h}(\bar{\boldsymbol{x}}^l) + \nabla_x \boldsymbol{h}(\bar{\boldsymbol{x}}^l)(\boldsymbol{x} - \bar{\boldsymbol{x}}^l) + \boldsymbol{C}\boldsymbol{y}) \end{array}\right\} \forall l \in \bar{\mathbf{F}} \\ & \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} \in \boldsymbol{Y} \in \{0,1\}^q \\ & \sum\limits_{i \in \mathbf{B}^l} y_i - \sum\limits_{i \in \mathbf{N}^l} y_i \leq |\mathbf{B}^l| - 1 \quad \forall l \in \mathbf{F} \end{array}$$

Obtain the solution and $\boldsymbol{y}^{k+1}$

If the solution to the master is greater than the current lower bound

Update the lower bound.

If $UBD - LBD \leq \epsilon$

Terminate

Else

Update the counter: $k = k + 1$

Go to step 2

This algorithm is not guaranteed to determine the global solution unless certain convexity conditions are met.

3.4 Outer Approximation/Equality Relaxation/Augmented Penalty

The **OA/ER/AP** algorithm [VG90] is a modification of the **OA/ER** algorithm. The objective of this algorithm is to avoid convexity assumptions necessary for finding the global solution using the **OA/ER** algorithm. This algorithm addresses the same class of problems as **OA/ER**:

$$
\begin{array}{lrcl}
\min\limits_{\boldsymbol{x},\boldsymbol{y}} & f(\boldsymbol{x}) + \boldsymbol{c}^T\boldsymbol{y} & & \\
\text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{y} & \leq & \mathbf{0} \\
 & \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{C}\boldsymbol{y} & = & \mathbf{0} \\
 & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
 & \boldsymbol{y} & \in & \boldsymbol{Y} \in \{0,1\}^q
\end{array}
\tag{21}
$$

This algorithm uses a relaxation of the linearizations of the master problem in order to expand the feasible region. Through this expansion of the feasible region, the probability of cutting part of the feasible region due to one of the linearizations is reduced. Note that this does not guarantee the possible elimination of part of the feasible region and thus the determination of the global solution cannot be guaranteed.

The linearizations in the master problem are relaxed by including slack variables in the constraints. The violations of the linearizations are penalized by including weighted sums of the slack variables in the objective function.

The difference between the **OA/ER** and **OA/ER/AP** algorithms is in the master problem formulations. The **OA/ER/AP** master problem has the following formulation:

$$
\begin{array}{lll}
\min\limits_{\boldsymbol{x},\boldsymbol{y},s,\boldsymbol{p},\boldsymbol{q}} & \boldsymbol{c}^T\boldsymbol{y} + \mu + \sum\limits_l u_l s_l + \sum\limits_l \sum\limits_i v_{i,l} p_{i,l} + \sum\limits_l \sum\limits_i w_{i,l} q_{i,l} & \\
\text{s.t.} & \left.\begin{array}{rcl} \mu + s_l & \geq & f(\boldsymbol{x}^l) + \nabla_x f(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) \\ \boldsymbol{p}_l & \geq & \boldsymbol{g}(\boldsymbol{x}^l) + \nabla_x \boldsymbol{g}(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) + \boldsymbol{B}\boldsymbol{y} \\ \boldsymbol{q}_l & \geq & \boldsymbol{T}^l(\boldsymbol{h}(\boldsymbol{x}^l) + \nabla_x \boldsymbol{h}(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) + \boldsymbol{C}\boldsymbol{y}) \end{array}\right\} \forall l \in \mathbf{F} & \\
 & \left.\begin{array}{rcl} \boldsymbol{p}_l & \geq & \boldsymbol{g}(\bar{\boldsymbol{x}}^l) + \nabla_x \boldsymbol{g}(\bar{\boldsymbol{x}}^l)(\boldsymbol{x} - \bar{\boldsymbol{x}}^l) + \boldsymbol{B}\boldsymbol{y} \\ \boldsymbol{q}_l & \geq & \boldsymbol{T}^l(\boldsymbol{h}(\bar{\boldsymbol{x}}^l) + \nabla_x \boldsymbol{h}(\bar{\boldsymbol{x}}^l)(\boldsymbol{x} - \bar{\boldsymbol{x}}^l) + \boldsymbol{C}\boldsymbol{y}) \end{array}\right\} \forall l \in \bar{\mathbf{F}} & \\
 & \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n & \\
 & \boldsymbol{y} \in \boldsymbol{Y} \in \{0,1\}^q & \\
 & s_l, \boldsymbol{p}_l, \boldsymbol{q}_l \geq 0 & \\
 & \sum\limits_{i \in \mathbf{B}^l} y_i - \sum\limits_{i \in \mathbf{N}^l} y_i \leq |\mathbf{B}^l| - 1 \quad \forall l \in \mathbf{F} &
\end{array}
\tag{22}
$$

where s^k is a slack scalar for iteration k and $\boldsymbol{p}_k$ and $\boldsymbol{q}_k$ are slack vectors at iteration k for the inequality and relaxed equality constraints respectively. The weights for the penalty terms, u_l, $v_{i,k}$, and $w_{i,k}$, are determined from the multipliers of the corresponding constraints from the solution of the primal problem. The correspondence between the constraints in the primal problem and the multipliers, μ_0^k, μ^k, and λ^k is as follows:

$$
\begin{array}{llcll}
\min\limits_{\boldsymbol{x}} & b^T\boldsymbol{y}^k + \mu_B & & & \\
\text{s.t.} & J(\boldsymbol{x}) - \mu_B & \leq & \mathbf{0} & \leftarrow \mu_0^k \\
 & g(\boldsymbol{x}) - c^T\boldsymbol{y}^k & \leq & \mathbf{0} & \leftarrow \mu^k \\
 & h(\boldsymbol{x}) - d^T\boldsymbol{y}^k & = & \mathbf{0} & \leftarrow \lambda^k
\end{array}
$$

The weights for the slack variables are assigned as follows ([VG90]):

$$
\begin{array}{ll}
u_k & = 1000\mu_0^k \\
v_{i,k} & = 1000\mu^k \\
w_{j,k} & = 1000\lambda^k
\end{array}
$$

3.4.1 OA/ER/AP Algorithm

Step 1

Obtain starting point: $\boldsymbol{y}^1$

Set counter: $k = 1$

Set the lower bound: $LBD = -\infty$

Set the upper bound: $UBD = +\infty$

Set the convergence tolerance: $\epsilon \geq 0$

Step 2

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

$$\begin{array}{lrcl} \min\limits_{x} & f(\boldsymbol{x}) + \boldsymbol{c}^T\boldsymbol{y}^k & & \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) & \leq & \boldsymbol{B}\boldsymbol{y}^k \\ & \boldsymbol{h}(\boldsymbol{x}) & = & \boldsymbol{C}\boldsymbol{y}^k \end{array}$$

Obtain the optimal $\boldsymbol{x}^k$ and the Lagrange multipliers μ^k

If the primal is feasible

If the optimal solution of the primal is less than UBD

Update the upper bound (UBD)

Else

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

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

Obtain the optimal $\bar{\boldsymbol{x}}^k$ and Lagrange multipliers $\bar{\mu}^k$

Step 3

Determine the matrix $\boldsymbol{T}$ as follows:

$\boldsymbol{T}^k = \text{diag}(t_{ii})$ where $t_{ii} = \text{sign}(\mu_i^k)$

Set the values for the penalty parameters, u, $\boldsymbol{v}$, and $\boldsymbol{w}$

Solve the relaxed master problem:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x},\boldsymbol{y},s,\boldsymbol{p},\boldsymbol{q}} & \boldsymbol{c}^T\boldsymbol{y} + \mu + \sum\limits_l u_l s_l + \sum\limits_l \sum\limits_i v_{i,l} p_{i,l} + \sum\limits_l \sum\limits_i w_{i,l} q_{i,l} \\ \text{s.t.} & \left.\begin{array}{rcl} \mu + s_l & \geq & f(\boldsymbol{x}^l) + \nabla_x f(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) \\ \boldsymbol{p}_l & \geq & \boldsymbol{g}(\boldsymbol{x}^l) + \nabla_x \boldsymbol{g}(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) + \boldsymbol{B}\boldsymbol{y} \\ \boldsymbol{q}_l & \geq & \boldsymbol{T}^l(\boldsymbol{h}(\boldsymbol{x}^l) + \nabla_x \boldsymbol{h}(\boldsymbol{x}^l)(\boldsymbol{x} - \boldsymbol{x}^l) + \boldsymbol{C}\boldsymbol{y}) \end{array}\right\} \forall l \in \mathbf{F} \\ & \left.\begin{array}{rcl} \boldsymbol{p}_l & \geq & \boldsymbol{g}(\bar{\boldsymbol{x}}^l) + \nabla_x \boldsymbol{g}(\bar{\boldsymbol{x}}^l)(\boldsymbol{x} - \bar{\boldsymbol{x}}^l) + \boldsymbol{B}\boldsymbol{y} \\ \boldsymbol{q}_l & \geq & \boldsymbol{T}^l(\boldsymbol{h}(\bar{\boldsymbol{x}}^l) + \nabla_x \boldsymbol{h}(\bar{\boldsymbol{x}}^l)(\boldsymbol{x} - \bar{\boldsymbol{x}}^l) + \boldsymbol{C}\boldsymbol{y}) \end{array}\right\} \forall l \in \bar{\mathbf{F}} \\ & \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} \in \boldsymbol{Y} \in \{0,1\}^q \\ & s_l, \boldsymbol{p}_l, \boldsymbol{q}_l \geq 0 \\ & \sum\limits_{i \in \mathbf{B}^l} y_i - \sum\limits_{i \in \mathbf{N}^l} y_i \leq |\mathbf{B}^l| - 1 \quad \forall l \in \mathbf{F} \end{array}$$

Obtain optimal $\boldsymbol{y}^{k+1}$

If the solution to the master is greater than the current lower bound

Update the lower bound

If $UBD - LBD \leq \epsilon$

Terminate

Else
Update the counter: $k = k + 1$
Go to step 2

3.5 Generalized Outer Approximation

The Generalized Outer Approximation, **GOA**, algorithm [FL94] generalizes the **OA** approach to handle the following class of MINLP problems:

$$\begin{array}{lrcl} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}, \boldsymbol{y}) & & \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) & \leq & \mathbf{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} & \in & \{0,1\}^q \end{array} \tag{23}$$

The difference between this formulation and that for the **OA** algorithm is that there is no restriction on the separability in the $\boldsymbol{x}$ and $\boldsymbol{y}$ variables and the linearity of the $\boldsymbol{y}$ variables.

The differences between the **GOA** algorithm and the **OA** algorithm are the treatment of infeasibilities, a new master problem formulation, and the unified treatment of exact penalty functions.

3.5.1 Primal Problem

The primal problem is formulated in the same manner as in **OA**. However, if the primal is infeasible, then the following feasibility problem is solved for $\boldsymbol{y} = \boldsymbol{y}^k$:

$$\begin{array}{lrcll} \min\limits_{\boldsymbol{x}} & \sum\limits_{i \in I'} w_i g_i^+(\boldsymbol{x}, \boldsymbol{y}^k) & & & \\ \text{s.t.} & \boldsymbol{g}_i(\boldsymbol{x}, \boldsymbol{y}^k) & \leq & \mathbf{0} & i \in I \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n & \end{array} \tag{24}$$

where I is the set of feasible inequality constraints and I' is the set of infeasible inequality constraints. With this formulation of the feasibility problem, the linearizations of the nonlinear inequality constraints about the solution of the feasibility problem are violated.

3.5.2 Master Problem

The master problem for **GOA** is formulated based on the **OA** ideas of projection onto the $\boldsymbol{y}$-space and the outer approximation of the objective function and feasible region. The difference in the master problem formulation is in the utilization of the infeasible primal information.

The projection onto the $\boldsymbol{y}$-space is the same as for **OA**:

$$\begin{array}{ll} \min\limits_{\boldsymbol{y}} & v(\boldsymbol{y}) \\ \text{s.t.} & \boldsymbol{y} \in \boldsymbol{Y} \cap \boldsymbol{V} \end{array} \tag{25}$$

where

$$v(\boldsymbol{y}) = \boldsymbol{c}^T\boldsymbol{y} + \begin{array}{ll} \inf\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{B}^T\boldsymbol{y} \leq \mathbf{0} \\ & \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \end{array} \tag{26}$$

and

$$\boldsymbol{V} = \{\boldsymbol{y} : \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{y} \leq 0, \ \text{ for some } \ \boldsymbol{x} \in \boldsymbol{X}\} \tag{27}$$

The formulation of the master problem follows from the outer approximation of the the problem $v(\boldsymbol{y})$ and a representation of the set $\boldsymbol{V}$. For the outer approximation of $v(\boldsymbol{y})$, the linearizations of the objective function and constraints are used. The set $\boldsymbol{V}$ is replaced by linearizations of the constraints at $\boldsymbol{y}^k$ for which the primal is infeasible, and the feasibility problem has solution $\boldsymbol{x}^k$. Thus, the master problem has the following formulation:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x},\boldsymbol{y},\mu_{\text{GOA}}} & \boldsymbol{c}^T\boldsymbol{y} + \mu_{\text{GOA}} \\ \text{s.t.} & \\ & \left. \begin{array}{rcl} \mu_{\text{GOA}} & \geq & f(\boldsymbol{x}^k,\boldsymbol{y}^k) + \nabla f(\boldsymbol{x}^k,\boldsymbol{y}^k) \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^k \\ \boldsymbol{y} - \boldsymbol{y}^k \end{pmatrix} \\ \mathbf{0} & \geq & \boldsymbol{g}(\boldsymbol{x}^k,\boldsymbol{y}^k) + \nabla \boldsymbol{g}(\boldsymbol{x}^k,\boldsymbol{y}^k) \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^k \\ \boldsymbol{y} - \boldsymbol{y}^k \end{pmatrix} \end{array} \right\} \forall k \in \mathbf{F} \\ & \mathbf{0} \geq \boldsymbol{g}(\boldsymbol{x}^k,\boldsymbol{y}^k) + \nabla \boldsymbol{g}(\boldsymbol{x}^k,\boldsymbol{y}^k) \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^k \\ \boldsymbol{y} - \boldsymbol{y}^k \end{pmatrix} \ \forall k \in \bar{\mathbf{F}} \\ & \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} \in \boldsymbol{Y} \in \{0,1\}^q \end{array} \tag{28}$$

where $\mathbf{F}$ is the set of all $\boldsymbol{y}^k$ such that the primal problem is feasible, and $\bar{\mathbf{F}}$ is the set of all $\boldsymbol{y}^k$ such that the primal problem is infeasible.

As in **OA**, relaxation and an iterative procedure are used to solve the problem since the solution of the complete master problem (28) requires all feasible and infeasible solutions of the primal problem. For the solution of the relaxed master problem, the known feasible and infeasible solutions are used for the outer approximation.

3.5.3 GOA Algorithm

Step 1

Obtain starting point: $\boldsymbol{y}^1$
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 $\boldsymbol{y}^k$:

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

If the primal is feasible
Obtain the optimal $\boldsymbol{x}^k$
Update the feasibility set: $\mathbf{F} = \mathbf{F} \cup k$
If the optimal solution of the primal is less than UBD
Update the upper bound (UBD)
Else
Solve the infeasible primal problem for the fixed values of $\boldsymbol{y}^k$:

$$\begin{array}{rrcl} \min\limits_{\boldsymbol{x}} & \sum\limits_{i \in I'} w_i g_i^+(\boldsymbol{x}, \boldsymbol{y}^k) & & \\ \text{s.t.} & \boldsymbol{g}_i(\boldsymbol{x}, \boldsymbol{y}^k) & \leq & \mathbf{0} \quad i \in I \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \end{array}$$

Obtain the optimal $\bar{\boldsymbol{x}}^k$
Update the feasibility set: $\bar{\mathbf{F}} = \bar{\mathbf{F}} \cup k$

Step 3

Solve the relaxed master problem:

$$
\begin{array}{ll}
\min\limits_{\boldsymbol{x},\boldsymbol{y},\mu_{\text{GOA}}} & \boldsymbol{c}^T\boldsymbol{y} + \mu_{\text{GOA}} \\
\text{s.t.} & \left.\begin{array}{rcl}
\mu_{\text{GOA}} & \geq & f(\boldsymbol{x}^l,\boldsymbol{y}^l) + \nabla f(\boldsymbol{x}^l,\boldsymbol{y}^l)\begin{pmatrix} \boldsymbol{x}-\boldsymbol{x}^l \\ \boldsymbol{y}-\boldsymbol{y}^l \end{pmatrix} \\
\mathbf{0} & \geq & \boldsymbol{g}(\boldsymbol{x}^l,\boldsymbol{y}^l) + \nabla \boldsymbol{g}(\boldsymbol{x}^l,\boldsymbol{y}^l)\begin{pmatrix} \boldsymbol{x}-\boldsymbol{x}^l \\ \boldsymbol{y}-\boldsymbol{y}^l \end{pmatrix}
\end{array}\right\} \forall l \in \mathbf{F} \\
& \mathbf{0} \geq \boldsymbol{g}(\bar{\boldsymbol{x}}^l,\boldsymbol{y}^l) + \nabla \boldsymbol{g}(\bar{\boldsymbol{x}}^l,\boldsymbol{y}^l)\begin{pmatrix} \boldsymbol{x}-\bar{\boldsymbol{x}}^l \\ \boldsymbol{y}-\boldsymbol{y}^l \end{pmatrix} \quad \forall l \in \bar{\mathbf{F}} \\
& \boldsymbol{x} \in \boldsymbol{X} \\
& \boldsymbol{y} \in \boldsymbol{Y}
\end{array}
$$

Obtain the solution and $\boldsymbol{y}^{k+1}$

If the solution to the master is greater than the current lower bound

Update the lower bound.

If $UBD - LBD \leq \epsilon$

Terminate

Else

Update the counter: $k = k + 1$

Go to step 2

3.6 Generalized Cross Decomposition

The Generalized Cross Decomposition, **GCD**, algorithm [Hol90] exploits the advantages of Dantzig-Wolfe Decomposition and **GBD** and simultaneously utilizes primal and dual information. This algorithm can address problems with the general MINLP formulation (2) where the constraints are partitioned into two sets:

$$
\begin{array}{lrcl}
\min\limits_{\boldsymbol{x},\boldsymbol{y}} & f(\boldsymbol{x},\boldsymbol{y}) & & \\
\text{s.t.} & \boldsymbol{g}_1(\boldsymbol{x},\boldsymbol{y}) & \leq & \mathbf{0} \\
& \boldsymbol{g}_2(\boldsymbol{x},\boldsymbol{y}) & \leq & \mathbf{0} \\
& \boldsymbol{h}_1(\boldsymbol{x},\boldsymbol{y}) & = & \mathbf{0} \\
& \boldsymbol{h}_2(\boldsymbol{x},\boldsymbol{y}) & = & \mathbf{0} \\
& \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{y} & \in & \{0,1\}^q
\end{array}
\tag{29}
$$

This algorithm consists of two phases and convergence tests. Phase I involves the solution of the primal and dual subproblems where the primal subproblem provides an upper bound on the solution along with Lagrange multipliers for the dual subproblem. The dual subproblem provides a lower bound on the solution of the problem and supplies new values of the $\boldsymbol{y}$ variables for the primal subproblem. Both the primal and dual subproblems provide cuts for the master problem (Phase II). In Phase II either a primal master problem or a Lagrange relaxation master problem is solved. The primal master problem is formulated using the same derivation as for the **GBD** master problem while the Lagrange relaxation master problem is formulated by using Lagrangian duality. The algorithm also uses several convergence tests to determine whether or not solutions of various subproblems can provide bound or cut improvement.

The algorithm for **GCD** is based on the idea that it is desirable to solve as few master problems as possible since these are generally more computationally intensive. The algorithm makes extensive use of the primal and dual subproblems to reduce the number of times the master problem must be solved to obtain the solution.

3.6.1 GCD Algorithm

Step 1

Obtain starting point: $\boldsymbol{y}^1$
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: $\bar{\mathbf{F}} = \emptyset$

Step 2

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

$$\begin{array}{rrcl} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}, \boldsymbol{y}^k) & & \\ \text{s.t.} & \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{y}^k) & \leq & \mathbf{0} \\ & \boldsymbol{g}_2(\boldsymbol{x}, \boldsymbol{y}^k) & \leq & \mathbf{0} \\ & \boldsymbol{h}_1(\boldsymbol{x}, \boldsymbol{y}^k) & = & \mathbf{0} \\ & \boldsymbol{h}_2(\boldsymbol{x}, \boldsymbol{y}^k) & = & \mathbf{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \end{array}$$

Obtain the optimal $\boldsymbol{x}^k$ and Lagrange multipliers λ_1^k, λ_2^k, μ_1^k, and μ_2^k
If the primal is feasible
  Update the feasibility set: $\mathbf{F} = \mathbf{F} \cup k$
  If the optimal solution of the primal is less than UBD
    update the upper bound (UBD)

Perform **CTD** test for λ_1^k and μ_1^k

Else

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

$$\begin{array}{lrcl}\min\limits_{\boldsymbol{x},\alpha} & \multicolumn{3}{l}{\alpha_{i1}+\alpha_{i2}+\alpha_{e1}^{+}+\alpha_{e1}^{-}+\alpha_{e2}^{+}+\alpha_{e2}^{-}} \\ \text{s.t.} & \boldsymbol{g}_1(\boldsymbol{x},\boldsymbol{y}^k)-\alpha_{i1} & \leq & \mathbf{0} \\ & \boldsymbol{g}_2(\boldsymbol{x},\boldsymbol{y}^k)-\alpha_{i2} & \leq & \mathbf{0} \\ & \boldsymbol{h}_1(\boldsymbol{x},\boldsymbol{y}^k)+\alpha_{e1}^{+}-\alpha_{e1}^{-} & = & \mathbf{0} \\ & \boldsymbol{h}_2(\boldsymbol{x},\boldsymbol{y}^k)+\alpha_{e2}^{+}-\alpha_{e2}^{-} & = & \mathbf{0} \\ & \alpha_{i1},\alpha_{i2},\alpha_{e1}^{+},\alpha_{e1}^{-},\alpha_{e2}^{+},\alpha_{e2}^{-} & \geq & 0\end{array}$$

Obtain the optimal $\bar{\boldsymbol{x}}^k$ and the Lagrange multipliers $\bar{\lambda}_1^k$, $\bar{\lambda}_2^k$, $\bar{\mu}_1^k$ and $\bar{\mu}_2^k$

Update the infeasibility set: $\bar{\mathbf{F}} = \bar{\mathbf{F}} \cup k$

Perform **CTDU** test for $\bar{\lambda}_1^k$ and $\bar{\mu}_1^k$

Step 3

If **CTD** or **CTDU** test from Step 2 is not passed

Solve Relaxed Lagrange Relaxation Master problem:

$$\begin{array}{lrcll}\max\limits_{\lambda_1,\mu_1,\mu_b} & \mu_b & & & \\ \text{s.t.} & \mu_b & \leq & f(\boldsymbol{x}^l,\boldsymbol{y}^l)+(\lambda_1)^T\boldsymbol{h}_1(\boldsymbol{x}^l,\boldsymbol{y}^l)+(\mu_1)^T\boldsymbol{g}_1(\boldsymbol{x}^l,\boldsymbol{y}^l) & l\in\mathbf{F} \\ & \mathbf{0} & \leq & (\lambda_1)^T\boldsymbol{h}_1(\boldsymbol{x}^l,\boldsymbol{y}^l)+(\mu_1)^T\boldsymbol{g}_1(\bar{\boldsymbol{x}}^l,\boldsymbol{y}_l) & l\in\bar{\mathbf{F}} \\ & \mu_1 & \geq & \mathbf{0} & \end{array}$$

Obtain optimal λ_1^k and μ_1^k

Step 4

If $k \in \mathbf{F}$ or the **CTD** or **CTDU** test from Step 2 is not passed

Solve the Dual Subproblem:

$$\begin{array}{lrcl}\min\limits_{\boldsymbol{x},\boldsymbol{y}} & \multicolumn{3}{l}{f(\boldsymbol{x},\boldsymbol{y})+(\lambda_1^k)^T\boldsymbol{h}_1(\boldsymbol{x},\boldsymbol{y})+(\mu_1^k)^T\boldsymbol{g}_1(\boldsymbol{x},\boldsymbol{y})} \\ \text{s.t.} & \boldsymbol{h}_2(\boldsymbol{x},\boldsymbol{y}) & = & \mathbf{0} \\ & \boldsymbol{g}_2(\boldsymbol{x},\boldsymbol{y}) & \leq & \mathbf{0}\end{array}$$

Obtain optimal $\boldsymbol{y}^{k+1}$

If the solution is greater than the lower bound

Update the lower bound

Else

Solve the Dual Feasibility Subproblem:

$$\begin{array}{lrcl}\min\limits_{\boldsymbol{x},\boldsymbol{y}} & \multicolumn{3}{l}{(\bar{\lambda}_1^k)^T\boldsymbol{h}_1(\boldsymbol{x},\boldsymbol{y})+(\bar{\mu}_1^k)^T\boldsymbol{g}_1(\boldsymbol{x},\boldsymbol{y})} \\ \text{s.t.} & \boldsymbol{h}_2(\boldsymbol{x},\boldsymbol{y}) & = & \mathbf{0} \\ & \boldsymbol{g}_2(\boldsymbol{x},\boldsymbol{y}) & \leq & \mathbf{0}\end{array}$$

Obtain the optimal $\boldsymbol{y}^{k+1}$

Step 5

Perform the **CTP** test for $\boldsymbol{y}^{k+1}$

If the **CTP** test fails

Solve the Relaxed Primal Master Problem:

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

Obtain the optimal $\boldsymbol{y}^{k+1}$

If the solution is greater than the lower bound

Update the lower bound

If $UBD - LBD \leq \epsilon$

Terminate

Else

Update the counter: $k = k + 1$

Go to step 2

The convergence tests are defined as follows:

CTP Test At the k^{th} iteration, if the solution from the Dual Subproblem $\boldsymbol{y}^{k+1}$ satisfies

$$\begin{array}{rcll} UBD & \geq & f(\boldsymbol{x}^l,\boldsymbol{y}^{k+1}) + (\lambda^l)^T \boldsymbol{h}(\boldsymbol{x}^l,\boldsymbol{y}^{k+1}) + (\mu^l)^T \boldsymbol{g}(\boldsymbol{x}^l,\boldsymbol{y}^{k+1}) & l \in \mathbf{F} \\ 0 & \geq & (\bar{\lambda}^l)^T \boldsymbol{h}(\bar{\boldsymbol{x}}^l,\boldsymbol{y}^{k+1}) + (\bar{\mu}^l)^T \boldsymbol{g}(\bar{\boldsymbol{x}}^l,\boldsymbol{y}^{k+1}) & l \in \bar{\mathbf{F}} \end{array}$$

then $\boldsymbol{y}^{k+1}$ will provide an upper bound or cut improvement. Otherwise, the Relaxed Primal Master is solved to obtain a new $\boldsymbol{y}^{k+1}$.

CTD Test At the k^{th} iteration, if the feasible solution of the Primal Problem satisfies:

$$LBD \leq f(\boldsymbol{x}^l,\boldsymbol{y}^l) + (\lambda_1^k)^T \boldsymbol{h}(\boldsymbol{x}^l,\boldsymbol{y}^l) + (\mu_1^k)^T \boldsymbol{g}(\boldsymbol{x}^l,\boldsymbol{y}^l) \quad l \in \mathbf{F}$$

then λ_1^k and μ_1^k will provide a lower bound or cut improvement. Otherwise, the Relaxed Lagrange Relaxation Master is solved to obtain new λ_1^k and μ_1^k.

CTDU Test At the k^{th} iteration, if the Primal Problem is infeasible and the solution to the Infeasible Primal Problem satisfies

$$0 \leq (\bar{\lambda}_1^k)^T \boldsymbol{h}(\bar{\boldsymbol{x}}^l,\boldsymbol{y}^l) + (\bar{\mu}_1^k)^T \boldsymbol{g}(\bar{\boldsymbol{x}}^l,\boldsymbol{y}^l) \quad l \in \bar{\mathbf{F}}$$

then $\bar{\lambda}_1^k$ and $\bar{\mu}_1^k$ will provide cut improvement. Otherwise, the Relaxed Lagrange Relaxation Master is solved to obtain λ_1^k and μ_1^k.

The situations where the **GCD** algorithm reduces to Dantzig-Wolfe decomposition and **GBD** can be observed. First, when the tests **CTD** and **CTDU** are not passed at all iterations and only the relaxed primal master problem is used, **GCD** reduces to **GBD**. On the other hand, if the test **CTP** is not used at all iterations and only the relaxed primal Lagrange problem is used, then **GCD** reduces to Dantzig-Wolfe decomposition.

The algorithm for **GCD** is well-illustrated by the use of a flow diagram such as that in Figure 3.

3.7 Branch-and-Bound Algorithms

A number of branch-and-bound algorithms have been proposed to identify the global optimum solution of problems which are convex in the $\boldsymbol{x}$-space and relaxed $\boldsymbol{y}$-space. [Bea77, GR85, OOM90, BM91, QG92]. These algorithms can also be used for nonconvex problems of the form (2) but their convergence to the global optimum solution can only be guaranteed for convex problems. A basic principle common to all these algorithms is the generation of valid lower bounds on the original MINLP through its relaxation to a continuous problem. In most algorithms, the continuous problem is obtained by letting binary variables take on any value between 0 and 1. In most algorithms, this relaxation is an NLP problem. The only exception is the algorithm of [QG92], discussed in Section 3.7.3, in which an LP problem is obtained. 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.

Although the size of the branch-and-bound tree is finite, and convergence is guaranteed, it is desirable to explore as few nodes of the tree as possible. The selection of the branching node and variable and the solution of the relaxed problem all affect the convergence characteristics of the algorithm. Several strategies have been suggested in the literature.

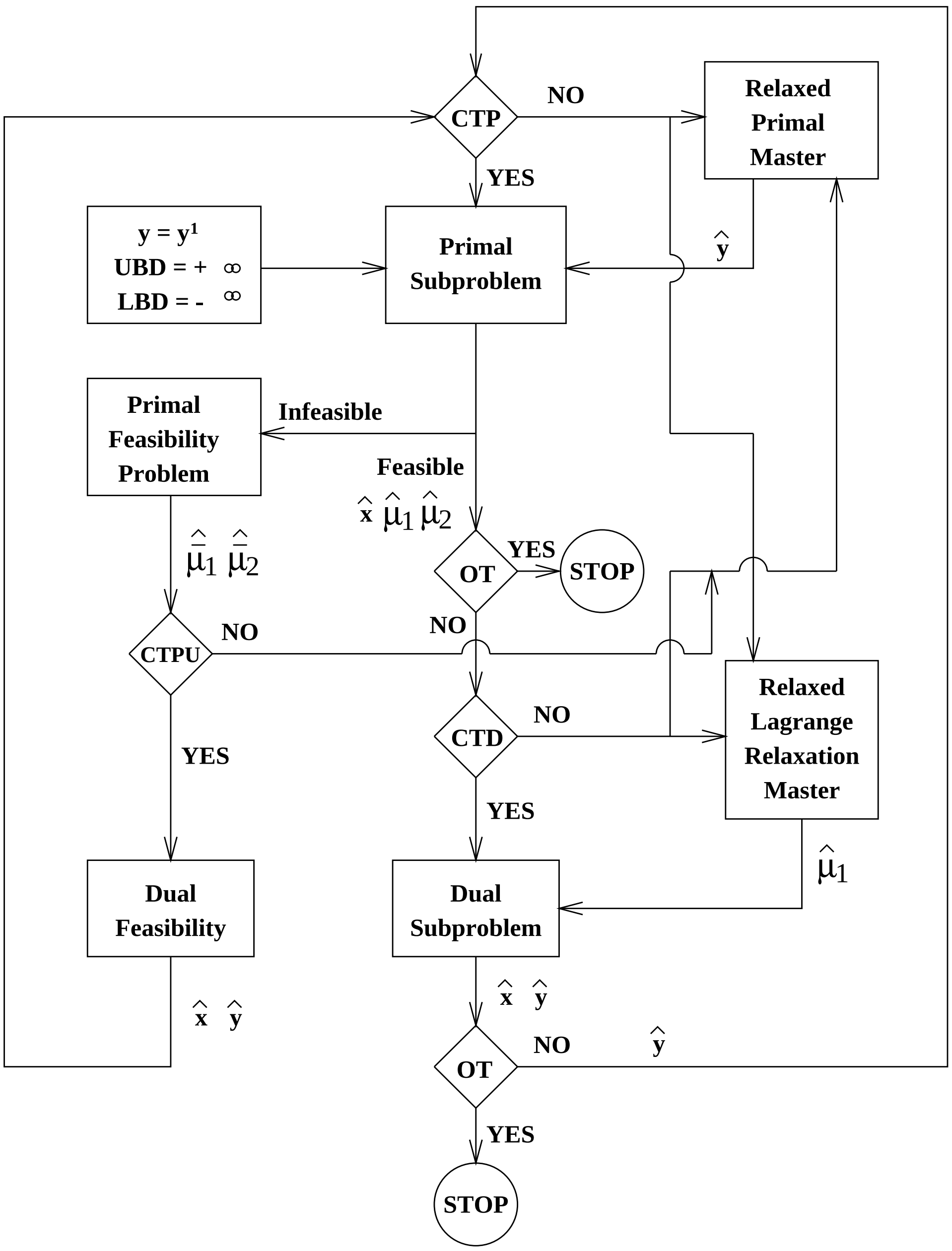

Figure 3: Generalized Cross Decomposition Flow Diagram

3.7.1 Selection of the Branching Node

Since all nodes whose lower bound is less than the best upper bound must be explored before convergence can be declared, most algorithms use the *node with the lowest lower bound* for branching. It is often the strategy which minimizes computational requirements [LW66].

In some cases, a *depth-first approach* has been adopted [GR85, QG92]. In this case, the last node created is selected for branching and the branch-and-bound tree is explored vertically, until an integer solution is obtained and backtracking can be used to move back towards the root of the tree, until a node with a child still open for search is identified. This strategy can lead to the generation of a tighter upper bound as levels in the branch-and-bound tree where a large fraction of the binary variables are fixed to one of their bounds are quickly reached. However, it may result in the solution of an unnecessarily large number of relaxations.

The *breadth-first approach* can also be followed in exploring the solution space. In this case, every node on a level is branched on before moving on to the next level. This approach can be especially useful at the initial levels of the tree, in order to identify promising branches that should then be explored through a depth-first approach. Thus, a combination of a depth-first and breadth-first strategies is likely to result in smaller computational expense than the adoption of either a single one of these techniques.

Finally, a node can be selected based on the "quality" of the solution of the relaxation, as measured by the *estimation of the node* [GR85]. In this case, nodes for which the values of the integer variables at the solution of the continuous relaxation lie far away from an integer solution are penalized. This deviation from integrality is combined with the value of the objective function for each node, to yield a quantity referred to as estimation. The node with the lowest estimation is then chosen as the next branching node.

3.7.2 Selection of the Branching Variable

The most commonly used criterion for the selection of a branching variable y^B is the *most fractional variable rule* [GR85, OOM90]. The variable which is farthest from its binary bounds at the solution of the node to be explored is selected for branching.

Other approaches attempt to determine which binary variable has the greatest effect on the lower bound of the problem. If the user has prior knowledge of the problem, *branching priorities* may be set to accelerate the increase of the best lower bound [GR85]. The quantitative analysis of the effect of each binary variable has also been proposed through the use of *pseudo-costs* [BGG$^+$71, GR85]. The pseudo-cost of a variable y_j is defined by calculating the relative change in the objective function when y_j is fixed to 0 or 1. Thus, if the solution of the NLP with $0 \leq y_j \leq 1$ is $y_j = y_j^*$ and $f(x, y) = f^*$, and the optimum objective value with $y_j = 0$ is f_0, the lower pseudo-cost associated with y_j is $PC_j^L = (f_0 - f^*)/y_j^*$. Similarly, if f_1 is the optimum objective value for $y_j = 1$, the upper pseudo-cost is defined as $PC_j^U = (f_1 - f^*)/(1 - y_j^*)$. To avoid excessive computational requirements, pseudo-costs are only calculated once. At each node, the pseudo-costs of fractional variables are updated by computing the minimum of $PC_j^L y_j^*$ and $PC_j^U(1 - y_j^*)$, where y_j^* is the value of y_j at the solution of the NLP at the branching node. The variable with the maximum pseudo-cost is selected for branching as it is expected to lead to the greatest increase in the lower bound on the problem.

3.7.3 Generation of a Lower Bound

At each node, a relaxation of the MINLP problem must be solved in order to generate a lower bound. This procedure can be costly for large problems and [BM91] advocate the early detection of non-integer solutions in order to reduce the time spent solving NLPs. If the binary variables appear to be converging to values away from their bounds, the NLP solver should be interrupted before full convergence has been achieved and the current node should be branched on.

[QG92] suggest further relaxation of the problem to an LP in order to obtain lower bounds. When the lower bounding LP has an integer solution, this integer combination is used to formulate an NLP problem which yields an upper bound on the original problem. The LP problems are generated by linearizing the nonlinear functions at the solution of each NLP. In order to prevent the LPs from becoming excessively large, a reformulation which combines all nonlinearities into one inequality constraint is used. The first linearization is obtained by fixing the binary variables to an arbitrary combination and solving the resulting NLP.

3.7.4 Algorithmic Procedure

The general algorithmic statement for branch-and-bound approaches is 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, $J = \{0\}$;
Set relaxed $\boldsymbol{y}$-space, $\boldsymbol{Y}_0 = [0, 1]^q$.
$\boldsymbol{N}$, the list of nodes to be explored, is empty.

Step 2 *Bounding*

Solve relaxed NLP problem, for $j \in J$:

$$\begin{aligned} LBD_j = \min \quad & f(\boldsymbol{x}, \boldsymbol{y}) \\ \text{s.t.} \quad \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) &= 0 \\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) &\leq 0 \\ \boldsymbol{x} &\in \boldsymbol{X} \subseteq \mathbb{R}^n \\ \boldsymbol{y} &\in \boldsymbol{Y}_j \end{aligned}$$

If all $\boldsymbol{y}$ variables are integer at the solution:
If $LBD_j \leq UBD^*$, $UBD^* = LBD_j$.
Else, fathom the current node.
If some $\boldsymbol{y}$ variables are fractional:
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 *Branching*

Select the next node to be explored, N_i $(i < k)$, from list $\boldsymbol{N}$. Its lower bound is LBD_i and the corresponding $\boldsymbol{y}$-space is $\boldsymbol{Y}_i$.
Select a branching variable y^B.
Create two new regions $\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_i \cap \{\boldsymbol{y} | y^B = 0\}$ and $\boldsymbol{Y}_{k+2} = \boldsymbol{Y}_i \cap \{\boldsymbol{y} | y^B = 1\}$.
Set $J = k + 1, k + 2$ and $k = k + 2$. Go back to Step 2.

3.8 Extended Cutting Plane (ECP)

The cutting plane algorithm proposed by [Kel60] for NLP problems has been extended to MINLPs [WPG94, WP95]. This Extended Cutting Plane algorithm (ECP) can address problems of the form:

$$
\begin{array}{lrcl}
\min & \boldsymbol{c}_x^T \boldsymbol{x} + \boldsymbol{c}_y^T \boldsymbol{y} & & \\
\text{s.t.} & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) & \leq & 0 \\
& \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{y} & \in & \boldsymbol{Y} \text{integer}
\end{array}
\tag{30}
$$

where $\boldsymbol{c}_x$ and $\boldsymbol{c}_y$ are constant vectors.

Problems with a nonlinear objective function, $f(\boldsymbol{x},\boldsymbol{y})$, can be reformulated by introducing a new variable z such that $f(\boldsymbol{x},\boldsymbol{y}) - z \leq 0$ and minimizing z.

The ECP algorithm relies on the linearization of one of the nonlinear constraints at each iteration and the solution of the increasingly tight MILP made up of these linearizations. The solution of the MILP problem provides a new point on which to base the choice of the constraint to be linearized for the next iteration of the algorithm. Unlike the Outer Approximation, described in Section 3.2, the ECP does not require the solution of any NLP problems for the generation of an upper bound. As a result, a large number of linearizations are required for the approximation of highly nonlinear problems and the algorithm does not perform well in such cases. Due to the use of linearizations, convergence to the global optimum solution is guaranteed only for problems involving inequality constraints which are convex in the $\boldsymbol{x}$ and relaxed $\boldsymbol{y}$-space.

Given a point $(\boldsymbol{x}^0,\boldsymbol{y}^0)$ in the feasible set, the function $g_i(\boldsymbol{x},\boldsymbol{y})$ can be underestimated by

$$
g_i(\boldsymbol{x}^0,\boldsymbol{y}^0) + \left(\frac{\partial g_i}{\partial \boldsymbol{x}}\right)_{\boldsymbol{x}^0,\boldsymbol{y}^0} (\boldsymbol{x} - \boldsymbol{x}^0) + \left(\frac{\partial g_i}{\partial \boldsymbol{y}}\right)_{\boldsymbol{x}^0,\boldsymbol{y}^0} (\boldsymbol{y} - \boldsymbol{y}^0). \tag{31}
$$

The function $g_{j_0}(\boldsymbol{x},\boldsymbol{y})$, where $j_0 = \text{argmax}_i \; g_i(\boldsymbol{x}^0,\boldsymbol{y}^0)$, is then used to construct an underestimating MILP problem

$$
\begin{array}{lrcl}
\min & \boldsymbol{c}_x^T \boldsymbol{x} + \boldsymbol{c}_y^T \boldsymbol{y} & & \\
\text{s.t.} & l_0(\boldsymbol{x},\boldsymbol{y}) & \leq & 0 \\
& \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{y} & \in & \boldsymbol{Y} \text{integer}
\end{array}
\tag{32}
$$

where $l_0(\boldsymbol{x},\boldsymbol{y}) = g_{j_0}(\boldsymbol{x}^0,\boldsymbol{y}^0) + \left(\frac{\partial g_{j_0}}{\partial \boldsymbol{x}}\right)_{\boldsymbol{x}^0,\boldsymbol{y}^0} (\boldsymbol{x} - \boldsymbol{x}^0) + \left(\frac{\partial g_{j_0}}{\partial \boldsymbol{y}}\right)_{\boldsymbol{x}^0,\boldsymbol{y}^0} (\boldsymbol{y} - \boldsymbol{y}^0)$.

At any iteration k, a single constraint is chosen for linearization and the corresponding linearization $l_k(\boldsymbol{x}, \boldsymbol{y})$ is added to the MILP. All linear constraints from the original MINLP problem are, of course, incorporated into the MILP from the start of the algorithm.

3.8.1 Algorithmic Procedure

The algorithmic procedure is:

Step 1

Set absolute tolerance ϵ; set $LBD^* = -\infty$.
Set iteration counter $k = 0$ and select starting point $(\boldsymbol{x}^0, \boldsymbol{y}^0)$.

Step 2

Solve kth MILP problem:

$$\begin{array}{rrcl} LBD^* = \min & \boldsymbol{c}_x^T \boldsymbol{x} + \boldsymbol{c}_y^T \boldsymbol{y} & & \\ \text{s.t.} & l_i(\boldsymbol{x}, \boldsymbol{y}) & \leq & 0, i = 0, \cdots, k-1 \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} & \in & \boldsymbol{Y}_j \end{array}$$

The optimal solution of the MILP is at $(\boldsymbol{x}^k, \boldsymbol{y}^k)$.
Find $j_k = \text{argmax}_i \; g_i(\boldsymbol{x}^k, \boldsymbol{y}^k)$.

Step 3

If $g_{j_k}(\boldsymbol{x}^k, \boldsymbol{y}^k) \leq \epsilon$, convergence has been reached. Terminate with solution LBD^*.
Otherwise, proceed to Step 4.

Step 4

Construct $l_k(\boldsymbol{x}, \boldsymbol{y})$. Set $k = k + 1$. Return to Step 2.

3.9 Feasibility Approach

This algorithm was proposed by [MM85] as an extension to MINLP problems of the MINOS algorithm [MS93] for large-scale nonlinear problems.

The premise of their approach is that the problems to be treated are sufficiently large that techniques requiring the solution of several NLP relaxations, such as the branch-and-bound approach of Section 3.7, have prohibitively large costs. They therefore wish to account for the presence of the integer variables in the formulation and solve the mixed-integer problem directly. This is achieved by fixing most of the integer variables to one their bounds (the *nonbasic* variables) and allowing the remaining small subset (the *basic* variables) to take discrete values in order to identify feasible solutions. After each iteration, the reduced costs of the variables in the nonbasic set are computed to measure of their effect on the objective function. If a change causes the objective function to decrease, the appropriate variables are removed from the nonbasic set and allowed to vary for the next iteration. When no more improvement in the objective function is possible, the algorithm is terminated. This strategy leads to the identification of a local solution.

The basic and nonbasic sets are initialized through the solution of continuous relaxation of the NLP. The solution obtained must then be rounded to a feasible integer solution through a heuristic approach. The feasibility approach has been tested on two types of large-scale problems: quadratic assignment and gas pipeline network design. The second problem poses more difficulties as few variables are at either of their bounds when the continuous relaxation is solved. Therefore, the problem has relatively few nonbasic variables. This trend is preserved throughout the run, thus increasing the computational complexity of each iteration.

3.10 Logic Based Approach

An alternative to the direct solution of the MINLP problem was proposed by [TG96]. Their approach stems from the work of [KG89] on a modeling/decomposition strategy which avoids the zero-flows generated by the non-existence of a unit in a process network. The first stage of the algorithm is the reformulation of the MINLP into a generalized disjunctive program of the form:

$$
\begin{array}{lrcl}
\min & f(\boldsymbol{x}) + \sum\limits_i c_i & & \\
\text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) & \leq & 0 \\
& \left[\begin{array}{c} Y_i \\ \boldsymbol{h}^i(\boldsymbol{x}) \leq 0 \\ c_i = \gamma_i \end{array}\right] & \vee & \left[\begin{array}{c} \neg Y_i \\ B^i \boldsymbol{x} = 0 \\ c_i = 0 \end{array}\right] \; i \in D \\
& \Omega(\boldsymbol{Y}) & = & \text{True} \\
& \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{c} & \geq & 0 \\
& \boldsymbol{Y} & \in & \{\text{True, False}\}^q
\end{array}
\tag{33}
$$

where $\boldsymbol{c}$ is the variable vector representing fixed charges, $\boldsymbol{x}$ is the vector representing all other continuous variables (flowrates, temperatures, ...) and

$\boldsymbol{Y}$ is the vector of Boolean variables, which indicate the status of a disjunction (True or False) and are associated with the units in the network. The set of disjunctions D allows the representation of different configurations, depending on the existence of the units. $\Omega(\boldsymbol{Y})$ is the set of logical relationships between the Boolean variables, representing the interactions between different network units. Instead of resorting to binary variables within a single model, the disjunctions are used to generate a different model for each different network structure. Since all continuous variables associated with the non-existing units are set to 0 ($B^i\boldsymbol{x} = 0, \boldsymbol{c} = 0$), this representation helps to reduce the size of the problems to be solved.

Two algorithms are suggested by [TG96] in order to solve problems such as (33). They are modifications of the Outer Approximation and Generalized Benders Decomposition presented in Sections 3.2 and 3.1 respectively.

3.10.1 Logic-Based Outer Approximation

In the case of disjunctive programming, the primal problem is obtained simply by fixing all the Boolean variables to a combination k of True and False, yielding an NLP problem of the form:

$$
\begin{array}{ll}
\min & f(\boldsymbol{x}) + \sum\limits_i c_i \\
\text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) \leq 0 \\
& \left\{\begin{array}{c} \boldsymbol{h}^i(\boldsymbol{x}) \leq 0 \\ c_i = \gamma_i \end{array}\right. \quad \text{for } Y_i^k = \text{True} \\
& \left\{\begin{array}{c} B^i\boldsymbol{x} = 0 \\ c_i = 0 \end{array}\right. \quad \text{for } Y_i^k = \text{False} \\
& \boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{c} \geq 0
\end{array}
\tag{34}
$$

The master problem is a disjunctive linear program. If K NLP problems have been solved, the nonlinear part of the objective and the nonlinear inequality constraints which are not part of disjunctions are linearized at the K solutions. The nonlinear constraints appearing in the ith disjunction are linearized at the solutions of the NLPs belonging to the subset $K_L^i = \{k | Y_i^k = \text{True}, k = 1, \ldots, K\}$. The master problem is then expressed as:

$$
\begin{array}{llcl}
\min & \mu_{\mathrm{OA}} + \sum\limits_i c_i & & \\
\text{s.t.} & f(\boldsymbol{x}^k) + \nabla f(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) & \leq & \mu_{\mathrm{OA}},\ k = 1, \ldots, K \\
& \boldsymbol{g}(\boldsymbol{x}^k) + \nabla \boldsymbol{g}(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) & \leq & 0, \qquad k = 1, \ldots, K \\
& \left[\begin{array}{l} Y_i \\ \boldsymbol{h}^i(\boldsymbol{x}^k) + \nabla \boldsymbol{h}^i(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) \leq 0 \\ c_i = \gamma_i \end{array}\right] & \vee & \left[\begin{array}{c} \neg Y_i \\ B^i \boldsymbol{x} = 0 \\ c_i = 0 \end{array}\right] \ i \in D \\
& \Omega(\boldsymbol{Y}) & = & \text{True} \\
& \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{c} & \geq & 0 \\
& \boldsymbol{Y} & \in & \{\text{True, False}\}^q
\end{array}
\tag{35}
$$

This type of problem can be solved as a disjunctive problem [Bea90], or as an MILP [Bal85, RG94]. To ensure that all nonlinear disjunction constraints are present in the master problem at every iteration, several NLPs must be solved at the start of the algorithm. The structures to be optimized are chosen in such a way that each Boolean variable is True in at least one structure. A method for identifying the minimum number of combinations required to satisfy this condition has been developed [TG96].

The algorithmic procedure for the logic-based outer approximation algorithm is very similar to the original outer approximation algorithm:

Step 0

Formulate the generalized disjunctive program of the form (33).

Step 1

Set the counter: $k = 1$
Set the lower bound: $LBD = -\infty$
Set the upper bound: $UBD = +\infty$
Set the convergence tolerance: $\epsilon \geq 0$
Determine the minimum set of structures needed to obtain cuts for all constraints. Generate the corresponding NLPs.

Step 2

Solve the NLP(s) for the fixed Boolean variables.
Obtain the optimal $\boldsymbol{x}$ and $\boldsymbol{c}$ vectors.
If the optimal solution of the NLP is less than the current upper bound, update the upper bound.

Step 3

Construct and solve the master problem.
Obtain its solution and $\boldsymbol{Y}^{k+1}$
If the solution of the master is greater than the current lower bound
    Update the lower bound.
If $UBD - LBD \leq \epsilon$
    Terminate
Else
    Update the counter: $k = k + 1$
    Go to step 2

3.10.2 Logic-Based Generalized Benders Decomposition

The Generalized Benders Decomposition framework is not as readily adapted to disjunctive programming as the Outer Approximation. The master problem is generated according to the following scheme:

1. Construct a master problem, as was done for the logic-based OA algorithm (Problem (35)).
2. Transform the problem to an MILP.
3. Based on the values used for the Boolean variables in the previous NLPs, fix the binary variables: LPs are obtained.
4. Solve the LPs and obtain the values of the Lagrange multipliers for the constraints and the optimal continuous variables values.
5. Use these to construct an MILP problem: this is the master problem for the logic-based GBD.

Both algorithms identify the global solution of problems which are convex for all combinations of the Boolean variables.

4 Global Optimization for Nonconvex MINLPs

The algorithms discussed so far have a major limitation when dealing with nonconvex problems. While identification of the global solution for convex problems can be guaranteed, a local solution is often obtained for nonconvex problems. A number of algorithms that have been developed to address different types of nonconvex MINLPs are presented in this section.

4.1 Branch-and-reduce algorithm

[RS95] extended the scope of branch-and-bound algorithms to problems for which valid convex underestimating NLPs can be constructed for the non-convex relaxations. The range of application of the proposed algorithm encompasses bilinear problems and separable problems involving functions for which convex underestimators can be built [McC76, AK90]. Because the nonconvex NLP must be underestimated at each node, convergence can only be achieved if the continuous variables are branched on. A number of tests are suggested to accelerate the reduction of the solution space. They are summarized here.

Optimality Based Range Reduction Tests For the first set of tests, an upper bound U on the nonconvex MINLP must be computed and a convex lower bounding NLP must be solved to obtain a lower bound L. If a bound constraint for variable x_i, with $x_i^L \leq x_i \leq x_i^U$, is active at the solution of the convex NLP and has multiplier $\lambda_i^* > 0$, the bounds on x_i can be updated as follows:

1. If $x_i - x_i^U = 0$ at the solution of the convex NLP and $\kappa_i = x_i^U - \frac{U-L}{\lambda_i^*}$ is such that $\kappa_i > x_i^L$, then $x_i^L = \kappa_i$.
2. If $x_i - x_i^L = 0$ at the solution of the convex NLP and $\kappa_i = x_i^L + \frac{U-L}{\lambda_i^*}$ is such that $\kappa_i < x_i^U$, then $x_i^U = \kappa_i$.

If neither bound constraint is active at the solution of the convex NLP for some variable x_j, the problem can be solved by setting $x_j = x_j^U$ or $x_j = x_j^L$. Tests similar to those presented above are then used to update the bounds on x_j.

Feasibility Based Range Reduction Tests In addition to ensuring that tight bounds are available for the variables, the constraint underestimators are used to generate new constraints for the problem. Consider the constraint $g_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0$. If its underestimating function $\underline{g}_i(\boldsymbol{x}, \boldsymbol{y}) = 0$ at the solution of the convex NLP and its multiplier is $\mu_i^* > 0$, the constraint

$$\underline{g}_i(\boldsymbol{x}, \boldsymbol{y}) \geq -\frac{U-L}{\mu_i^*}$$

can be included in subsequent problems.

The branch-and-reduce algorithm has been tested on very small problems.

4.2 Interval Analysis Based Algorithm

An algorithm based on interval analysis was proposed by [VEH96] to solve to global optimality problems of the form (2) with a twice-differentiable objective function and once-differentiable constraints. Interval arithmetic allows the computation of guaranteed ranges for these functions [Moo79, RR88, Neu90]. Although the algorithm is not explicitly described as a branch-and-bound approach, it relies on the same basic concepts of successive partitioning of the solution space and bounding of the objective function within each domain. Branching is performed on the discrete and the continuous variables. The main difference with the branch-and-bound algorithms described in Section 3.7 is that bounds on the problem solution in a given domain are not obtained through optimization. Instead, they are based on the range of the objective function in the domain under consideration, as computed with interval arithmetic. As a consequence, these bounds may be quite loose and efficient fathoming techniques are required in order to enhance convergence. [VEH96] suggest a number of tests to determine whether the optimal solution lies in the current domain. In addition, they propose branching strategies based on local solutions to the problem. In order to avoid combinatorial problems, integrality requirements for the discrete variables are removed when performing interval calculations. Convergence is declared when best upper and lower bounds are within a pre-specified tolerance and when the width of the corresponding region is below a pre-specified tolerance.

4.2.1 Node Fathoming Tests for Interval Algorithm

The *upper-bound test* is a classical criterion used in all branch-and-bound schemes: if the lower bound for a node is greater than the best upper bound for the MINLP, the node can be fathomed.

The *infeasibility test* is also used by all branch-and-bound algorithms. However, the identification of infeasibility using interval arithmetic differs from its identification using optimization schemes. Here, an inequality constraint $g_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0$ is declared infeasible if $G_i(\boldsymbol{X}, \boldsymbol{Y})$, its inclusion over the current domain, is positive. As soon as a constraint is found to be infeasible, the current node is fathomed.

The *monotonicity test* is only used in interval-based approaches. If a region is feasible, the monotonicity properties of the objective function can be tested. For this purpose, the inclusions of the gradients of the objective with respect to each variable are evaluated. If all the gradients have a constant sign for the current region, the objective function is monotonic and only one point needs to be retained from the current node.

The *nonconvexity test* is used to test the existence of a solution (local or global) within a region. If such a point exists, the Hessian matrix of the objective function at this point must be positive semi-definite. A sufficient condition for this to occur is the non-negativity of at least one of the diagonal elements of its interval Hessian matrix. The interval Hessian matrix is the inclusion of a Hessian matrix computed for a given domain.

[VEH96] advocate two additional tests to accelerate the fathoming process. The first is the so-called *lower bound test*. It requires the computation of a valid lower bound on the objective function through a method other than interval arithmetic. If the upper bound at a node is less than this lower bound, the region can be eliminated. The generation of such an upper bound may occur in an interval-based approach as the constraints are not used when evaluating the objective. Thus, a region may be found feasible because of the overestimation inherent in interval calculations, and have an upper bound lower than the optimal solution. For general problems, the computation of a valid and tight lower bound on the objective function requires the use of rigorous convex lower bounding techniques such as those described in Section 4.5.

The second test, the *distrust-region method*, aims to help the algorithm identify infeasible regions so that they can be removed from consideration. Based on the knowledge of an infeasible point, interval arithmetic is used to identify an infeasible hypercube centered on that point.

4.2.2 Branching Step

The variable with the widest range is selected for branching. It can be a continuous or a discrete variable. In order to determine where to split the chosen variable, a relaxation of the MINLP is solved locally.

Continuous Branching Variable If the optimal value of the continuous branching variable, x^*, is equal to one of the variable bounds, branch at the midpoint of the interval. Otherwise, branch at $x^* - \beta$, where β is a very small scalar.

Discrete Branching Variable If the optimal value of the continuous branching variable, y^*, is equal to the upper bound on the variable, define a region with $y = y^*$ and one with $y^L \leq y \leq y^* - 1$, where y^L is the lower bound on y. Otherwise, create two regions $y^L \leq y \leq int(y^*)$ and $int(y^*) + 1 \leq y \leq y^U$, where y^U is the upper bound on y.

This algorithm has been tested on a small example problem and a molecular design problem [VEH96].

4.3 Extended Cutting Plane for Nonconvex MINLPs

The use of the ECP algorithm for nonconvex MINLP problems was suggested in [WPG94], using a slightly modified algorithmic procedure. The main changes occur in the generation of new constraints for the MILP at each iteration (Step 4). In addition to the construction of the linear function $l_k(\boldsymbol{x}, \boldsymbol{y})$ at iteration k, the following steps are taken:

1. Remove all constraints for which $l_i(\boldsymbol{x}^k, \boldsymbol{y}^k) > g_{j_i}(\boldsymbol{x}^k, \boldsymbol{y}^k)$. These correspond to linearizations which did not underestimate the corresponding nonlinear constraint at all points due to the presence of nonconvexities.
2. Replace all constraints for which $l_i(\boldsymbol{x}^k, \boldsymbol{y}^k) = g_{j_i}(\boldsymbol{x}^k, \boldsymbol{y}^k) = 0$ by their linearization around $(\boldsymbol{x}^k, \boldsymbol{y}^k)$.
3. If constraint i is such that $g_i(\boldsymbol{x}^k, \boldsymbol{y}^k) > 0$, add its linearization around $(\boldsymbol{x}^k, \boldsymbol{y}^k)$.

The convergence criterion is also modified. In addition to the test used in Step 3, the following two conditions must be met:

1. $(\boldsymbol{x}^k - \boldsymbol{x}^{k-1})^T(\boldsymbol{x}^k - \boldsymbol{x}^{k-1}) \leq \delta$, a pre-specified tolerance.
2. $\boldsymbol{y}^k - \boldsymbol{y}^{k-1} = 0$.

The ECP algorithm has been used to address a nonlinear pump configuration problem [WPG94], where it was found to give good results for convex one-level problems, and to perform poorly for nonconvex problems. It has also been tested on a small convex MINLP from [DG86]. Finally, a comparative study between the Outer Approximation, the Generalized Benders Decomposition and the Extended Cutting Plane algorithm was presented in [SHW$^+$96]. A parameter estimation problem from FTIR spectroscopy and a purely integer problem were addressed.

4.4 Reformulation/Spatial Branch-and-Bound Algorithm

A global optimization algorithm branch-and-bound algorithm has been proposed in [SP97]. It can be applied to problems in which the objective and constraints are functions involving any combination of binary arithmetic operations (addition, subtraction, multiplication and division) and functions that are either concave over the entire solution space (such as ln) or convex over this domain (such as exp).

The algorithm starts with an automatic reformulation of the original nonlinear problem into a problem that involves only linear, bilinear, linear fractional, simple exponentiation, univariate concave and univariate convex terms. This is achieved through the introduction of new constraints and variables. The reformulated problem is then solved to global optimality using a branch-and-bound approach. Its special structure allows the construction of a convex relaxation at each node of the tree. The integer variables can be handled in two ways during the generation of the convex lower bounding problem. The integrality condition on the variables can be relaxed to yield a convex NLP which can then be solved globally. Alternatively, the integer variables can be treated directly and the convex lower bounding MINLP can be solved using a branch-and-bound algorithm as described in Section 3.7. This second approach is more computationally intensive but is likely to result in tighter lower bounds on the global optimum solution.

In order to obtain an upper bound for the optimum solution, several methods have been suggested. A local MINLP algorithm as those described in Section 3 can be used. The MINLP can be transformed to an equivalent nonconvex NLP by relaxing the integer variables. For example, a variable $y \in \{0,1\}$ can be replaced by a continuous variable $z \in [0,1]$ by including the constraint $z - z \cdot z = 0$. The nonconvex NLP is then solved locally to provide an upper bound. Finally, the discrete variables could be fixed to some arbitrary value and the nonconvex NLP solved locally.

Branching variables in this algorithm can be either continuous or discrete variables. An "approximation error" is computed for each term in the problem as the distance between the original term and its convex relaxation. A variable that participates in the term with the largest such error is selected for branching. Finally, the authors perform bound updates on all variables in order to ensure tight underestimators are generated. This algorithm has been applied to several problems such as reactor selection, distillation column design, nuclear waste blending, heat exchanger network design and multilevel pump configuration.

4.5 The SMIN-αBB Algorithm

This algorithm, proposed in [AAF7a], is designed to address the following class of problems to global optimality:

$$
\begin{array}{rrcl}
\min & f(\boldsymbol{x}) + \boldsymbol{x}^T A_0 \boldsymbol{y} + \boldsymbol{c}_0^T \boldsymbol{y} & & \\
\text{s.t.} & \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{x}^T A_1 \boldsymbol{y} + \boldsymbol{c}_1^T \boldsymbol{y} & = & 0 \\
& \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{x}^T A_2 \boldsymbol{y} + \boldsymbol{c}_2^T \boldsymbol{y} & \leq & 0 \\
& \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\
& \boldsymbol{y} & \in & \boldsymbol{Y} \text{integer}
\end{array}
\tag{36}
$$

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 (36) 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) described in Section 3.2. Otherwise, the Generalized Benders Decomposition (GBD) can be used, as discussed in Section 3.1, 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 (36) 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 pre-specified 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.

4.5.1 Convex Underestimating MINLP Generation

In order to transform an MINLP problem of the form (36) 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

$$\boldsymbol{L}(\boldsymbol{x}) = nt(\boldsymbol{x}) - \sum_i \alpha_i (x_i^U - x_i)(x_i - x_i^L) \tag{37}$$

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.

4.5.2 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].

4.5.3 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 all 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.

4.5.4 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 *Bounding*
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 *Branching*

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.

4.6 The GMIN-αBB algorithm

This algorithm operates within a classical branch-and-bound framework as described in Section 3.7. The main difference with the algorithms of [GR85], [OOM90] and [BM91] is its ability to identify the global optimum solution of a much larger class of problems of the form

$$\begin{array}{rrcl} \min\limits_{\boldsymbol{x},\boldsymbol{y}} & f(\boldsymbol{x},\boldsymbol{y}) & & \\ \text{s.t.} & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) & = & \boldsymbol{0} \\ & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) & \leq & \boldsymbol{0} \\ & \boldsymbol{x} & \in & \boldsymbol{X} \subseteq \mathbb{R}^n \\ & \boldsymbol{y} & \in & \mathcal{N}^q \end{array} \tag{38}$$

where $\mathcal{N}$ is the set of non-negative integers and the only condition imposed on the functions $f(\boldsymbol{x},\boldsymbol{y})$, $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y})$ and $\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y})$ is that their continuous relaxations possess continuous second-order derivatives.

This increased applicability results from 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 4.5.

At each node of the branch-and-bound tree, the nonconvex MINLP is relaxed to give a nonconvex NLP, which is then solved with the αBB algorithm. This 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 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.

The GMIN-αBB algorithm selects the node with the lowest lower bound for branching at every iteration. The branching variable selection strategy combines several approaches: branching priorities can be specified for some of the integer variables. When no variable has a priority greater than all other variables, the solution of the continuous relaxation is used to identify either the most fractional variable or the least fractional variable for branching.

Other strategies have been implemented to ensure a satisfactory convergence rate. In particular, bound updates on the integer variables can be performed at each level of the branch-and-bound tree. These can be carried out through the use of interval analysis. An integer variable, y^*, is fixed at its lower (or upper) bound and the range of the constraints is evaluated with interval arithmetic, using the bounds on all other variables. If the range of any constraint is such that this constraint is violated, the lower (or upper) bound on variable y^* can be increased (or decreased) by one. Another strategy for bound updates is to relax the integer variables, to convexify and underestimate the nonconvex constraints and to minimize (or maximize) a variable y^* in this convexified feasible region. The resulting lower (or upper) bound on relaxed variable y^* can then be rounded up (or down) to the nearest integer to provide an updated bound for y^*.

A number of small nonconvex MINLP test problems as well as the pump configuration problem of [WPG94] have been solved using this strategy.

5 Implementation: MINOPT

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 earliest implementations make 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, a framework **MINOPT** [SF97b] has been developed for the solution of general mathematical programming problems. The primary motivation for the development of **MINOPT** (Mixed Integer Nonlinear OPTimizer) was to provide a user friendly interface for the solution of MINLPs. The development has expanded to include an interface for solving many classes of problems which include both algebraic and differential models. The next section describes this package in more detail and includes the results of its application to a number of example problems.

MINOPT has been developed as a framework for the solution of various classes of optimization problems. Its development has been brought about by the particular 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** 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.

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)	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**

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

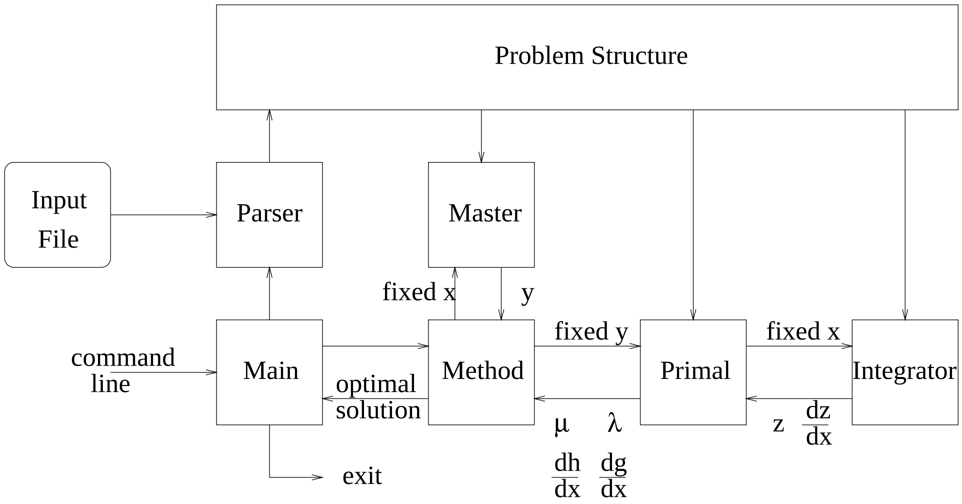

Figure 4: Program flow for **MINOPT**

- auto-initialization procedure—relaxed MINLP solved to determine the 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 4. 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.

6 Computational Studies

There are numerous MINLP problems that arise in process synthesis and design. **MINOPT** has been used as a computational tool to solve a wide variety of these problems including heat exchanger network synthesis problems, design and scheduling of multipurpose batch plants, reactor network synthesis, multicommodity facility location-allocation problems, parameter estimation, and distillation sequencing problems. The computational results for these problems run on a Hewlett Packard 9000/780 (C-160) are shown in Table 2.

Two computational examples are selected from the area of process synthesis. Both of these examples illustrate the use of a superstructure, the mathematical modeling of the superstructure as well as the implementation of an appropriate algorithm to solve the problem. The first problem is a distillation sequencing problem and the second is a heat exchanger network synthesis problem.

6.1 Distillation Sequencing

The distillation sequencing problem is to determine the configuration of a separation system which will separate a given feed stream into desired products which meet desired specifications. The details for the problem description and model formulation can be found in [AF90].

The input flowrate and composition are given along with the desired product flowrates and compositions. The problem is to determine the flowrates and compositions of the streams and the interconnection of distillation units which minimizes the annualized cost. The superstructure postulating two distillation units for one input and two outputs is shown in Figure 5.

The continuous variables for the problem are the flowrates, F, the mole fractions, x, and the recoveries for the light key and heavy key, r^{lk} and r^{hk}. The binary variables represent the existence of a distillation column, y.

Table 2: Computational Results

Problem	X	Y	Z	L	N	T	ITER	CPU TIME
gbd_test	1	3	–	4	1	D	2/2/2	0.06/0.06/0.06
oaer_test	6	3	–	5	2	D	3/2/3	0.18/0.25
ap_test	2	1	–	3	1	D	2/-/2	0.07/-/0.07
minutil	206	–	–	87	–	A		0.09
minmatch	82	18	–	129	–	B		0.11
plan	22	–	–	14	–	A		0.05
schedule	12	16	–	27	–	B		0.13
batdes	10	9	–	18	2	D	5/2/2	0.28/0.12/0.13
complex	8	3	–	10	–	B		0.08
alky	10	–	–	3	5	C		1.8
cart	6	–	4	3	4	E		0.63
param	12	–	2	1	10	E		4.38
feedtray	93	9	–	34	62	D	2/2/2	1.43/1.74/5.38
procsel	7	3	–	6	2	D	6/4/3	0.31/0.27/0.28
alan	4	4	–	7	1	D	8/4/6	0.20/0.12/0.17
facility1	16	6	–	18	1	D	3/3/4	0.14/0.17/0.21
facility2	54	12	–	33	1	D	5/5/7	0.43/0.73/0.97
ciric1	5	1	–	6	1	C	15/-/-	0.29/-/-
duran86-1	3	3	–	4	3	D	4/4/4	0.27/0.28/0.24
duran86-2	6	5	–	11	4	D	8/4/3	1.16/0.50/0.40
duran86-3	9	8	–	19	5	D	14/4/7	3.24/0.66/1.30
duran86-4	5	25	–	6	26	D	69/2/11	42.1/0.4/38.50
kocis88-4a	16	6	–	60	2	D	16/-/-	3.74/-/-
kocis88-4b	22	24	–	72	2	D	31/4/3	20.28/1.86/0.89
kocis89-2a	27	2	–	29	6	D	3/3/2	0.54/0.57/0.50
kocis89-2b	27	8	–	34	1	D	3/-/-	0.55
meanv1	21	14	–	44	1	D	10/4/3	0.48/0.33/0.26
meanv2	28	14	–	51	1	D	7/2/3	0.35/0.21/0.28
nous1	22	28	–	49	1	D	12/-/-	2.38/-/-
vdv	22	14	4	27	8	F	2/-/-	16.02/-/-
bindis	72	60	114	67	4	F	4/3/-	2860.0/2100.0/-

X, Y, Z, L, N and T indicate the number of x, y, and z variables, number of linear and nonlinear constraints and the type of the problem (A–LP, B–MILP, C–NLP, D–MINLP, E–NLP/DAE, F–MINLP/DAE). ITER and CPU TIME indicate the number of iterations and cpu time for the MINLP problems (GBD/OAER/OAERAP).

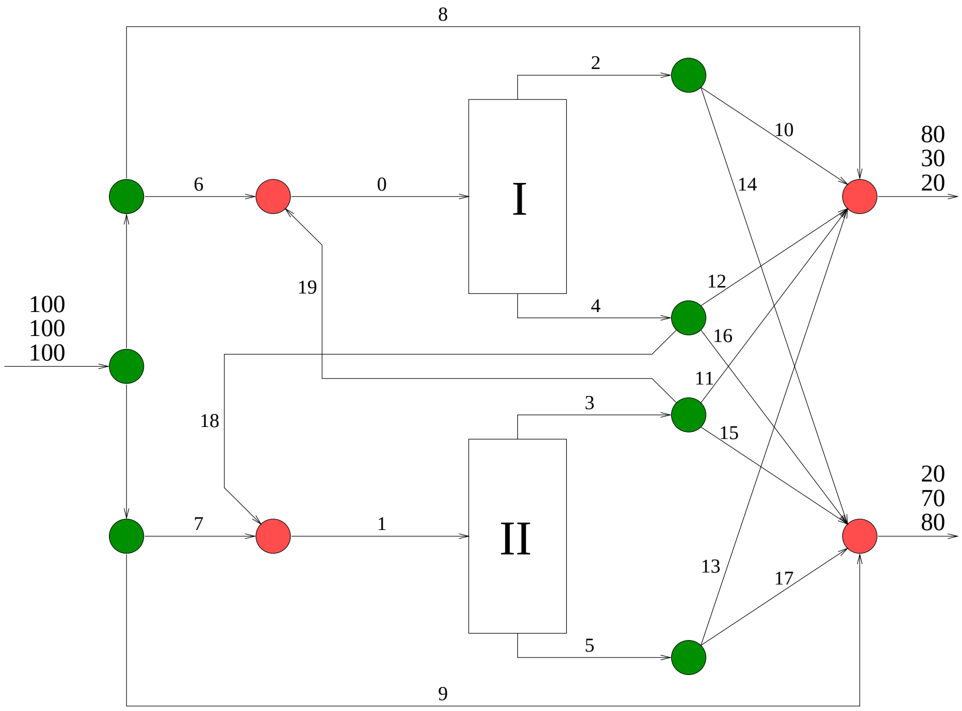

Figure 5: Superstructure for Nonsharp Separation System

The derivation of the mathematical model involves a number of indices and sets. The index set $I = \{i\}$ denotes components, $N = \{k\}$ denotes streams, $J = \{j\}$ denote columns, $S = \{s\}$ denote splitters, $M^c = \{m^c\}$ denote the mixers prior to each column, $M^f = \{m^f\}$ denote final mixers prior to each product, and $P = \{p\}$ denote the products. The splitter $s^0 \in S$ represents the initial splitting point of the feed stream. The following sets are defined for the connection of the sets of splitters and mixers with the streams in the superstructure.

$$
\begin{array}{lcl}
S_s^{in} & \equiv & \{l | l \in N \text{ is an inlet to splitter } s\} \; s \in S \\
S_s^{out} & \equiv & \{l | l \in N \text{ is an outlet from splitter } s\} \; s \in S \\
M_{m^f}^{in} & \equiv & \{l | l \in N \text{ is an inlet to mixer } m^f\} \; m^f \in M^f \\
M_{m^c}^{out} & \equiv & \{l | l \in N \text{ is an outlet to mixer } m^c\} \; m^c \in M^c
\end{array}
$$

The inlet and outlet streams of the column are given by the following:

$$
\begin{array}{lll}
SU_j & \equiv & \{n | n \in N \text{ is the inlet to column } j\} \; j \in J \\
SU_j^{\text{top}} & \equiv & \{p | p \in N \text{ is the top product of column } j\} \; j \in J \\
SU_j^{\text{bot}} & \equiv & \{q | q \in N \text{ is the bottom product of column } j\} \; j \in J
\end{array}
$$

The key components for the column are given by the following:

$$
\begin{array}{lll}
LK_j & \equiv & \{i | i \in I \text{ is the light key for column } j\} \; j \in J \\
HK_j & \equiv & \{i | i \in I \text{ is the heavy key for column } j\} \; j \in J \\
LHK_j & \equiv & \{i | i \in I \text{ is lighter than the heavy key for column } j\} \; j \in J \\
HLK_j & \equiv & \{i | i \in I \text{ is heavier than the light key for column } j\} \; j \in J \\
ND_{ik} & \equiv & \{i | i \in I \text{ is not present in stream } k\} \; i \in I, k \in N
\end{array}
$$

The problem formulation from [AF90] follows.

$$
\begin{array}{lll}
\min & \sum_{j \in J} \left\{ a_{0j} + \left(a_{1j} + a_{2j} r_{ij}^{lk} + a_{3j} r_{i'j}^{hk} + \sum_{i \in I} b_{ij} x_{ik} \right) F_k \right\} & \\
 & & i \in LK_j, i' \in HK_j, k \in SU_j \\
\text{s.t.} & \sum_{k \in S_s^{in}} F_k - \sum_{k \in S_s^{out}} F_k = 0 & s \in S \\
 & F_p x_{ik} - r_{ij}^{lk} f_{in} = 0 & i \in LK_j, n \in SU_j, p \in SU_j^{\text{top}}, j \in J \\
 & F_q x_{ik} - r_{ij}^{hk} f_{in} = 0 & i \in LK_j, n \in SU_j, q \in SU_j^{\text{bot}}, j \in J \\
 & f_{in} - F_n x_{in} = 0 & i \in I, n \in SU_j \\
 & f_{in} - F_p x_{ip} - F_q x_{iq} = 0 & i \in I, j \in J, n \in SU_j, \\
 & & p \in SU_j^{\text{top}}, q \in SU_j^{\text{bot}} \\
 & \sum_{l \in M_{m^c}^{in}} F_l x_{il} - \sum_{l \in M_{m^c}^{out}} f_{il} = 0 & i \in I, m^c \in M^c \\
 & x_{ik} = 0 & (i,k) \in ND_{ik} \\
 & \sum_{l \in M_{m^f}^{in}} -C_{ip} = 0 & i \in I, p \in P \\
 & \sum_{i \in I} x_{ik} = 1 & k \in S_s^{in} \\
 & F_k - \boldsymbol{U} y_j \leq 0 & j \in J, k \in SU_j \\
 & F_n - F_p - F_q = 0 & n \in SU_j, p \in SU_j^{\text{top}}, q \in SU_j^{\text{bot}}
\end{array}
$$

$$
\begin{array}{ll}
\sum_{l \in M_{m^c}^{in}} F_l - \sum_{l \in M_{m^c}^{out}} F_l = 0 & m^c \in M^c \\
\sum_{i \in M_{m^f}^{in}} F_l - \sum_i C_{ip} & p \in P, i \in I \\
\sum_i f_{il} - F_l & l \in SU_j \\
F_l - \sum_{i \in LHK_j} f_{in} - \sum_{i \in HK_j} [1 - (r_{ij}^{hk})^L] f_{in} \leq 0 & \\
 & n \in SU_j, l \in SU_j^{\text{top}} \\
F_l - \sum_{i \in HLK_j} f_{in} - \sum_{i \in LK_j} [1 - (r_{ij}^{lk})^L] f_{in} \leq 0 & \\
 & n \in SU_j, l \in SU_j^{\text{bot}} \\
F_l \geq 0 f_{in} \geq 0 & i \in I, n \in N, n \in SU_j \\
x_{ik} \geq 0 & i \in I, k \in (SU_j \cup SU_j^{\text{top}} \cup SU_j^{\text{bot}}) \\
(r^{lk})^L \leq r_{ij}^{lk} \leq (r_{ij}^{lk})^U & i \in LK_j, j \in J \\
(r^{hk})^L \leq r_{ij}^{hk} \leq (r_{ij}^{hk})^U & i \in HK_j, j \in J
\end{array}
$$

The data for the problem are taken from Example 2 in [AF90].

The nonlinearities in the problem are due to bilinear terms. Some of the continuous variables in the problem are partitioned as $\boldsymbol{y}$ variables along with the binary variables such that the primal problem is linear and thus convex. Since the $\boldsymbol{y}$ variables consist of both continuous and binary variables, the **GBD** algorithm must be used.

The problem is solved using **MINOPT** which incorporates a radial search algorithm into the **GBD** algorithm. This option specifies that the full NLP problem with only the binary variables fixed is solved after each primal problem. The algorithm converges in 3 iterations and takes 1.36 seconds of CPU time on a Hewlett Packard 9000/780 (C-160). The optimal sequence utilizes a single column and the optimal flowsheet is shown in Figure 6.

6.2 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

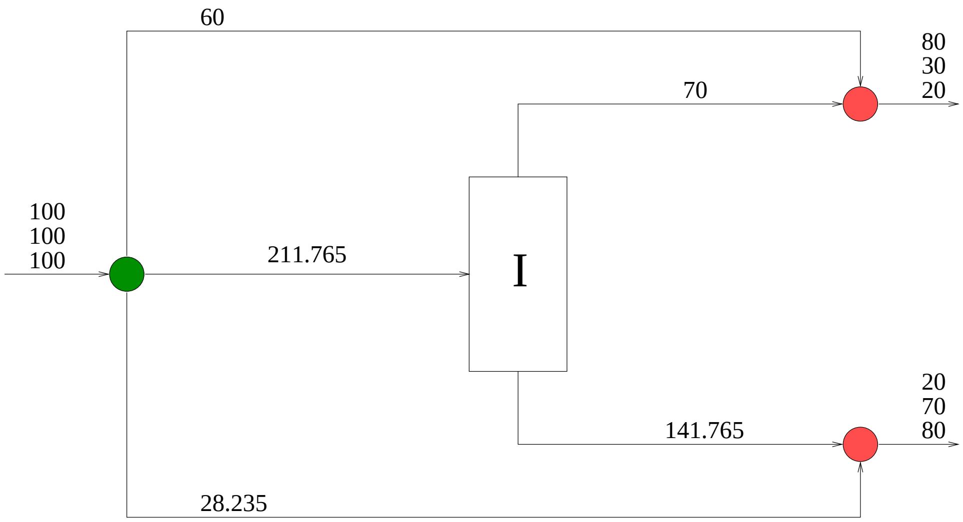

Figure 6: Optimal Flowsheet for the Nonsharp Separation Problem

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 4.5.

The stream data for the problem are summarized in Table 3. 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.

The superstructure for this problem is shown in Figure 7. There are 12 possible matches and therefore 12 binary variables. The global optimum configuration involves six heat exchangers and is shown in Figure 8. Given the set ST of K temperature locations, the set HP of hot process streams and the set CP of cold process streams, the general problem formulation is as follows:

$$
\begin{aligned}
\min & \sum_{i\in HP} C_{CU}Q_{CU,i} + \sum_{j\in CP} C_{HU}Q_{HU,j} \\
& + \sum_{i\in HP}\sum_{j\in CP}\sum_{k\in ST} CF_{ij}z_{ijk} + \sum_{i\in HP} CF_{i,CU}z_{CU,i} + \sum_{j\in CP} CF_{j,HU}z_{HU,j} \\
& + \sum_{i\in HP}\sum_{j\in CP}\sum_{k\in ST} \frac{C_{ij}Q_{ijk}}{U_{ij}\left[\Delta T_{ijk}\Delta T_{ijk+1}\left(\Delta T_{ijk}+\Delta T_{ijk+1}\right)/2\right]^{\frac{1}{3}}} \\
& + \sum_{i\in HP} \frac{C_{i,CU}Q_{CU,i}}{U_{CU,i}\left[\Delta T_{CU,i}\left(T_{out,i}-T_{in,CU}\right)\left(\Delta T_{CU,i}+T_{out,i}-T_{in,CU}\right)/2\right]^{\frac{1}{3}}} \\
& + \sum_{j\in CP} \frac{C_{j,HU}Q_{HU,j}}{U_{HU,j}\left[\Delta T_{HU,j}\left(T_{in,HU}-T_{out,j}\right)\left(\Delta T_{HU,j}+T_{in,HU}-T_{out,j}\right)/2\right]^{\frac{1}{3}}}
\end{aligned}
\tag{39}
$$

$$
\begin{aligned}
(T_{in,i}-T_{out,i})\,Fcp_i &= \sum_{k\in ST}\sum_{j\in CP} Q_{ijk} + Q_{CU,i} && \forall\, i\in HP \\
(T_{out,j}-T_{in,j})\,Fcp_j &= \sum_{k\in ST}\sum_{i\in HP} Q_{ijk} + Q_{HU,j} && \forall\, j\in CP \\
(T_{i,k}-T_{i,k+1})\,Fcp_i &= \sum_{j\in CP} Q_{ijk} && \forall\, k\in ST,\ \forall\, i\in HP \\
(T_{j,k}-T_{j,k+1})\,Fcp_j &= \sum_{i\in HP} Q_{ijk} && \forall\, k\in ST,\ \forall\, j\in CP
\end{aligned}
$$

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

Table 3: Stream data for heat exchanger network problem.

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	—

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 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.

Due to the linear participation of the binary variables, the problem can be solved locally using the Outer Approximation or Generalized Benders Decomposition algorithms described in Sections 3.2 and 3.1, and globally using the SMIN-αBB algorithm of Section 4.5.

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

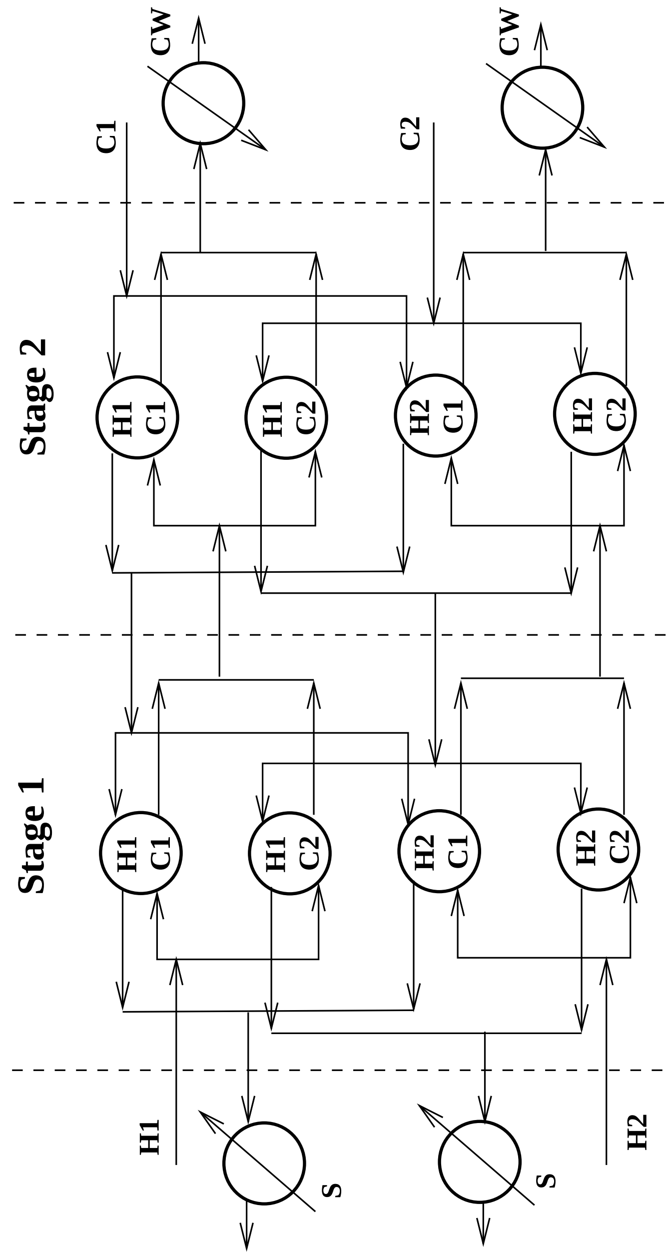

Figure 7: Superstructure for the heat exchanger network problem.

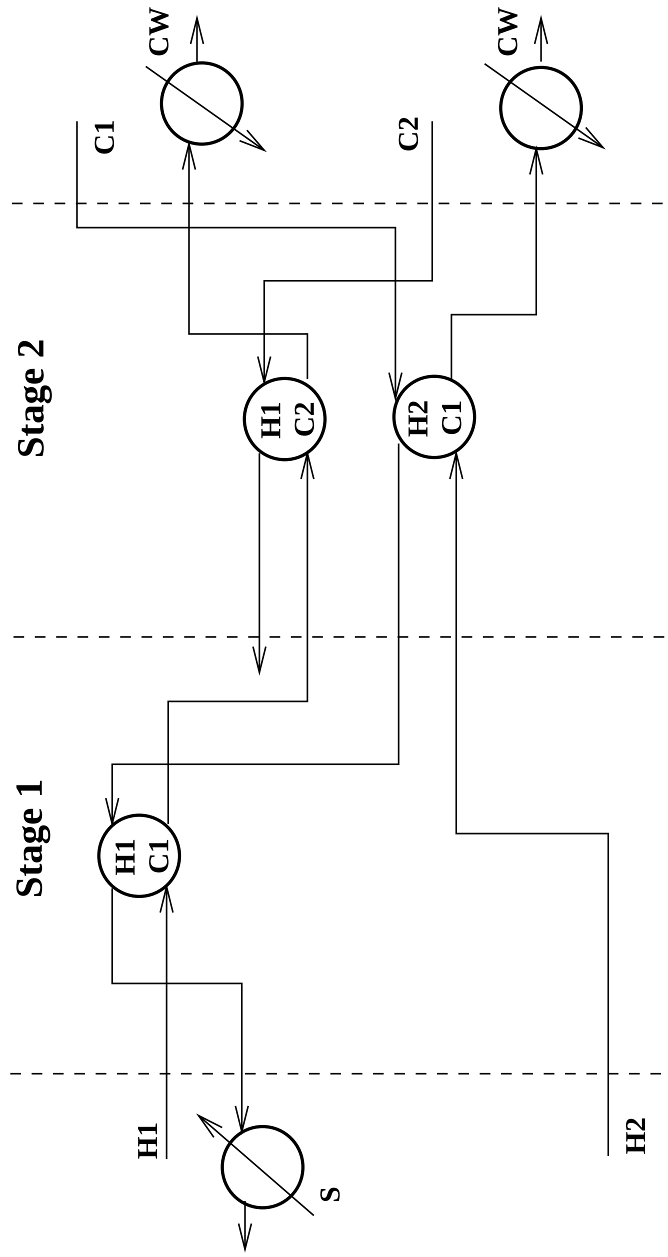

Figure 8: Optimum configuration for the heat exchanger network problem.

set to their lower bounds. The results of these runs are shown in Table 4. 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 random starting values are used for both the binary and continuous variables, the global optimum is obtained in all 30 runs.

Table 4: Local solutions for Heat Exchanger Network Synthesis problem obtained with **MINOPT**

GBD		
Local Solutions	number of times obtained	average number of iterations
154997	11	16
155510	18	18
161010	1	14

OAER		
Local Solutions	number of times obtained	average number of iterations
154997	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

6.2.1 Solution Strategy with the SMIN-αBB algorithm

When using the SMIN-αBB algorithm, the area-dependent cost of the heat exchangers must be underestimated using the general convex lower bounding function (37), 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 region. For the heat exchanger between hot stream i and cold stream j, the convex underestimator is expressed as

$$\begin{aligned}
&\frac{C_{ij} Q_{ijk}}{U_{ij}\left[\Delta T_{ijk}\Delta T_{ijk+1}\left(\Delta T_{ijk}+\Delta T_{ijk+1}\right)/2\right]^{\frac{1}{3}}} \\
&\qquad - \alpha_{ijk}^{Q}(Q_{ijk}^{U} - Q_{ijk})(Q_{ijk} - Q_{ijk}^{L}) \\
&\qquad - \alpha_{ijk}^{\Delta T_k}(\Delta T_{ijk}^{U} - \Delta T_{ijk})(\Delta T_{ijk} - \Delta T_{ijk}^{L}) \\
&\qquad - \alpha_{ijk}^{\Delta T_{k+1}}(\Delta T_{ijk+1}^{U} - \Delta T_{ijk+1})(\Delta T_{ijk+1} - \Delta T_{ijk+1}^{L}).
\end{aligned} \tag{40}$$

where α_{ijk}^{Q}, $\alpha_{ijk}^{\Delta T_k}$ and $\alpha_{ijk}^{\Delta T_{k+1}}$ are non-negative scalars obtained through one of the methods described by [AAF7a]. 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 $Q_{ijk} = Q_{ijk}^{L} = Q_{ijk}^{U} = 0$. 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 (40). Even though unit the area of unit (ijk) is 0, its cost appears in the underestimating objective function as

$$\begin{aligned}
-\alpha_{ijk}^{\Delta T_k}&(\Delta T_{ijk}^{U} - \Delta T_{ijk})(\Delta T_{ijk} - \Delta T_{ijk}^{L}) \\
&- \alpha_{ijk}^{\Delta T_{k+1}}(\Delta T_{ijk+1}^{U} - \Delta T_{ijk+1})(\Delta T_{ijk+1} - \Delta T_{ijk+1}^{L}).
\end{aligned} \tag{41}$$

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

$$\begin{aligned}
\alpha_{ijk}^{*,\Delta T_k} &= \alpha_{ijk}^{\Delta T_k}\, z_{ijk}^{U} \\
\alpha_{ijk}^{*,\Delta T_{k+1}} &= \alpha_{ijk}^{\Delta T_{k+1}}\, z_{ijk}^{U}.
\end{aligned} \tag{42}$$

where z_{ijk}^{U} is the current upper bound on variable z_{ijk}. According to Equation (42), if z_{ijk} is fixed to 0, its upper bound z_{ijk}^{U} is 0 and $\alpha_{ijk}^{*,\Delta T_k}$ and

$\alpha_{ijk}^{*,\Delta T_{k+1}}$ 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 4.5.2 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 9 and Table 5. 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.

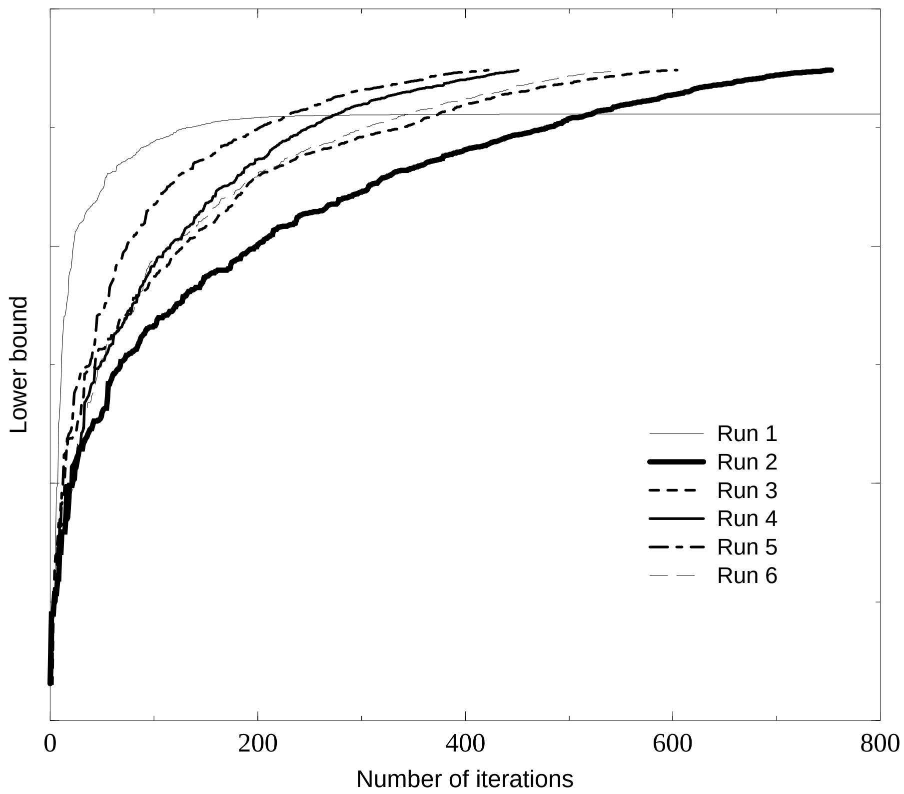

Figure 9: Progress of the lower bound for the heat exchanger network

Table 5: Global optimization of heat exchanger network – Note that Run 1 converges asymptotically.

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

6.2.2 Specialized Algorithm for Heat Exchanger Network Problems

A global optimization algorithm specifically designed for this type of problem was proposed in [ZG97]. The basic framework of this approach is a branch-and-bound algorithm where the branching variables are the stream temperatures. Special convex underestimators have been devised for the cost function, provided there is no stream splitting. Upper bounds on the problem are obtained by fixing the binary variables and solving a nonconvex NLP locally or globally. The branch-and-bound search is preceded by a heuristic local MINLP optimization step which allows the identification of a good starting point. No computational results using this approach are known for the example presented here.

7 Conclusions

As was demonstrated in this paper, mathematical programming techniques are a valuable tool for the solution of process 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 number of algorithms capable of addressing such problems were presented, including decomposition-based methods, branch-and-bound and cutting plane techniques. 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.

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 for 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, approaches to address important challenges such as the inclusion of dynamic models and optimal control problems into the MINLP framework are emerging [SF97a].

8 Acknowledgments

The authors gratefully acknowledge financial support from the National Science Foundation, the Air Force Office of Scientific Research, the National Institutes of Health, and Mobil Technology Company.

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.

[AF90] A. Aggarwal and C.A. Floudas, *Synthesis of general distillation sequences—nonsharp separations*, Comput. Chem. Eng. **14** (1990), no. 6, 631–653.

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

[AK90] F. A. Al-Khayyal, *Jointly constrained bilinear programs and related problems : An overview*, Comput. Math. Applic. **19** (1990), no. 11, 53–62.

[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.

[Bal85] E. Balas, *Disjunctive programming and a hierarchy of relaxations for discrete optimization problems*, SIAM Journal on Algebraic and Discrete Methods **6** (1985), 466–486.

[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.

[Bea90] N. Beaumont, *An algorithm for disjunctive programs*, European Journal of Operations Research **48** (1990), no. 3, 362–371.

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

[BGG$^+$71] M. Benichou, J. M. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and O. Vincent, *Experiments in mixed-integer linear programming*, Math. Prog. **1** (1971), no. 1, 76–94.

[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.

[Kel60] J. E. Kelley, *The cutting plane method for solving convex programs*, Journal of the SIAM **8** (1960), no. 4, 703–712.

[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.

[LW66] E. L. Lawler and D. E. Wood, *Branching and bound methods: A survey*, Oper. Res. (1966), no. 14, 699–719.

[McC76] G. P. McCormick, *Computability of global solutions to factorable nonconvex programs : Part I – convex underestimating problems*, Math. Prog. **10** (1976), 147–175.

[MM85] H. Mawengkang and B. A. Murtagh, *Solving nonlinear integer programs with large-scale optimization software*, Annals of Operations Research **5** (1985), no. 6, 425–437.

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

[Moo79] R. E. Moore, *Interval analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

[MS93] Bruce A. Murtagh and Michael A. Saunders, *Minos 5.4 user's guide*, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1993, Technical Report SOL 83-20R.

[Neu90] A. Neumaier, *Interval methods for systems of equations*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1990.

[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.

[RR88] H. Ratschek and J. Rokne, *Computer methods for the range of functions*, Ellis Horwood Series in Mathematics and its Applications, Halsted Press, 1988.

[RS95] H. S. Ryoo and N. V. Sahinidis, *Global optimization of nonconvex NLPs and MINLPs with applications in process design*, Comput. Chem. Eng. **19** (1995), no. 5, 551–566.

[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.

[SHW+96] H. Skrifvars, I. Harjunkoski, T. Westerlund, Z. Kravanja, and R. Pörn, *Comparison of different MINLP methods applied on certain chemical engineering problems*, Comput. Chem. Eng. Suppl. **20** (1996), S333–S338.

[SP97] E.M.B. Smith and C.C. Pantelides, *Global optimisation of nonconvex minlps*, Comput. Chem. Eng. **21** (1997), S791–S796.

[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.

[ZG97] J.M. Zamora and I.E. Grossmann, *A comprehensive global optimization approach for the synthesis of heat exchanger networks with no stream splits*, Comput. Chem. Eng. **21** (1997), S65–S70.