Mixed-Integer Optimization Techniques for the Design and Scheduling of Batch Processes

I.E. Grossmann, I. Quesada, R. Raman, V.T. Voudouris

EDRC 06-137-92

Mixed-Integer Optimization Techniques for the Design and Scheduling of Batch Processes

Ignacio E, Grossmann, Ignacio Quesada, Ramesh Raman and Vasilios T. Voudouris

Department of Chemical Engineering and Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.

Abstract: This paper provides a general overview of mixed-integer optimization techniques that are relevant for the design and scheduling of batch processes. A brief review of the recent application of these techniques in batch processing is first presented. The paper then concentrates on general purpose methods for mixed-integer linear (MELP) and mixed-integer nonlinear programming (MINLP) problems. Basic solution methods as well as recent developments are presented. A discussion on modelling and reformulation is also given to highlight the importance of this aspect in mixed-integer programming. Finally, several examples are presented in various areas of application to illustrate the performance of various methods.

Keywords: mathematical programming, mixed-integer linear programming, mixed-integer nonlinear programming, branch and bound, nonconvex optimization, reformulation techniques, batch design and scheduling

Introduction

The design, planning and scheduling of batch processes is a very fertile area for the application of mixed-integer programming techniques. The reason for this is that most of the mathematical optimization models that arise in these problems involve both discrete and continuous variables that must satisfy a set of equality and inequality constraints, and that must be chosen so as to optimize a given objective function. While there has been the recognition that many batch processing problems can be posed as mixed-integer optimization problems, the more extensive application of these techniques has only taken place in the recent past

It is the purpose of this paper to provide an overview of mixed-integer optimization techniques. We will first present a brief review of the application of these techniques in batch processing. We then provide a brief introduction to mixed-integer programming in order to determine a general classification of major problem types. Next we concentrate in both mixed-integer linear (MILP) and mixed-integer nonlinear programming (MINLP) techniques, introducing first the basic methods and then the recent developments that have taken place. We then present a discussion on modelling and reformulation, and finally, some numerical examples and results in various areas of application.

Review of applications

In this section we will present a brief overview of the application of mixed-integer programming in batch processing. More extensive reviews can be found in Papageorgaki and Reklaitis (1990c) and Reklaitis (1989,1991).

Mixed-integer nonlinear programming techniques have been applied mostly to design problems. Based on the problem considered by Sparrow et al (1975), Grossmann and Sargent (1979) were the first to formally model the design of multiproduct batch plants with parallel units and with single product campaigns as an MINLP problem. These authors showed that if one relaxes the numbers of parallel units to be continuous, the associated NLP corresponds to a geometric program that has a unique solution. Rather than solving the problem directly as an MINLP, the authors proposed a heuristic rounding scheme for the number of parallel units using nonlinear constraints based on the solution of the relaxed NLP. Since this problem provides a valid lower bound to the cost, optimality was established within the deviation of the rounded solution. This MINLP model was subsequently extended by Knopf et al (1982) in order to handle semi-continuous units. A further extension, was the MINLP model for a special type of multipurpose plants by Suhami and Mah (1982) in which simultaneous production was only allowed if products did not require the same processing stages. This model was subsequently modified by Vaselenak et al (1987) and by Faqir and Karimi (1989) to embed the selection of production campaigns. However, all these works did not rigorously solve the MINLP, but they relied on the rounding scheme by Grossmann and Sargent (1978) for obtaining an integer number of parallel units.

The first design application in which an MINLP model was rigorously solved was the work by Vaselenak et al (1987) who considered the retrofit design of multiproduct batch plants. These authors applied the outer-approximation method by Duran and Grossmann (1986) with a modification to handle separable nonconvex terms in the

objective. Recently, Fletcher et al (1991) removed the assumptions of equal volume for units operating out of phase by Vaselenak et al (1987), and formulated a new MINLP model that again was solved by the outer-approximation method. Also, Kocis and Grossmann (1989) formulated the MINLP model by Grossmann and Sargent (1978) in terms of 0-1 variables for the parallel units and solved it rigorously with the outer-approximation method as implemented in DICOPT. Subsequently, Wellons and Reklaitis (1989) applied this computer code to an MINLP model for multiproduct plants under uncertainty with staged expansions.

An important limitation in all the above applications was that convexity of the relaxed MINLP problem was a major requirement. Also, it became apparent that the solution of larger design problems could become expensive. The first difficulty was partially circumvented with the augmented penalty version of the outer-approximation algorithm proposed by Viswanathan and Grossmann (1990) and which was implemented in the computer code DICOPT++. This code was applied by Birewar and Grossmann (1990) for the simultaneous synthesis, sizing and scheduling of multiproduct batch plants which gives rise to a nonconvex MINLP model.

Papageorgaki and Reklaitis (1990a,b) developed a comprehensive MINLP model for the design multipurpose batch plants which involved nonconvex terms. They found that the code DICOPT would get trapped into suboptimal solutions and that the computation time was high. For this reason they proposed a special decomposition method in which the subproblems are NLFs with fixed 0-1 variables and campaign lengths and the master problem corresponds to a simplified MILP. Faqir and Karimi (1989) also modelled a special class of multipurpose batch plants with multiple production routes and discrete sizes as an MINLP problem that involves nonconvexities in the form of bilinear constraints. These authors proposed valid underestimators for these constraints and reduced the design problem to a sequence of MILP problems. Recently, Voudouris and Grossmann (1992a) have shown that several batch design problems, convex and nonconvex, can in fact be reformulated as MILP problems when they involve discrete sizes. Examples include the design of multiproduct batch plants with single product campaigns and the design of multipurpose batch plants with multiple routes. Finally, Ravemark and Rippin (1991) have reported computational experience in solving a variety of batch design problems as MINLP problems using the computer code DICOPT++, while Straub and Grossmann (1992) have applied it in the optimization of flexibility of multiproduct batch plants.

As for scheduling and planning, there have been a large number of MILP models reported in the Operations Research literature. However, in chemical engineering the first major MILP model for batch scheduling was proposed by Rich and Prokopakis (1986) for

the case of multipurpose batch plants in which the products were preassigned to processing units. They used the computer code LINDO (Schrage, 1986) to solve this problem, and later extended it to handle the production of a product over several predefined sets of units (Rich and Prokopakis, 1987). Ku and Karimi (1988) developed an MILP model for selecting production sequences that minimize the makespan in multiproduct batch plants with one unit per stage. Their model, which can accomodate a variety of storage policies, was also solved with the computer code LINDO.

A very general approach to the scheduling of batch operations was proposed by Kondili et al (1988) in which they developed a state-task network representation to model batch operations with complex process network structures. By discretizing the time domain they posed their problem as a multiperiod MDLP model that has the flexibility of accomodating variable batch sizes, splitting and mixing of batches, finite, unlimited or no storage, various transfer policies and resource constraints. Furthermore, the model has the flexibility of assigning equipment to different tasks. Recently, Shah et al (1991) have been able to considerably tighten the LP relaxation for this problem and develop a special purpose branch and bound method with which these authors have been able to solve problems with more than one thousand 0-1 variables. These authors have also extended their MILP model to some design and planning problems (Shah and Pantelides, 1991).

For the case of the no-wait flowshop scheduling problem Miller and Pekny (1991) (see also Pekny and Miller, 1991) formulated the problem as an asymmetric traveling salesman problem (see Gupta, 1976). For this model they developed a parallel branch and bound method that was coupled with a matching algorithm for detecting Hamiltonian cycles. The specialized implementation of their algorithm has allowed them to solve problems to optimality with more than 10,000 batches, which effectively translates to problems with more than 20,000 constraints and 100,000,000 0-1 variables.

Finally, MINLP models for scheduling of multipurpose batch plants have been formulated by Wellons and Reklaitis (1991) to handle flexible allocation of equipment and campaign formations. Due to the large size of these problems, these authors developed a special decomposition strategy for their solution. Sahinidis and Grossmann (1991a) considered the cyclic scheduling of continuous multiproduct plants with parallel lines and formulated the problem as a large-scale MINLP problem. They developed a solution method based on Generalized Benders decomposition for which they were able to solve problems with up to 800 0-1 variables, 23,000 continuous variables and 3000 constraints.

In summary, what this brief review shows is that both MILP and MINLP techniques are playing an increasingly important role in the modelling and solution of batch processing problems. This review also shows the importance of exploiting the structure of

these problems for developing reasonably efficient solution methods. It should also be mentioned that while there might be the temptation to resort to simpler optimization approaches such as simulated annealing, mixed integer programming provides a rigorous and deterministic framework, although it is not always the easiest one to apply. On the other hand, many mixed-integer problems that were regarded as unsolvable 10 years ago are currently being solved to optimality with reasonable computing requirements due to advances in algorithms and increased computer power.

Mixed-integer programming

In its most general form a mixed-integer program corresponds to the optimization problem,

$$\begin{aligned} \text{tnin} \quad & Z = f(x,y) \\ \text{s.t.} \quad & h(x,y) = 0 \\ & g(x,y) \pounds O \\ & x \in R^n \quad y \in N_+^m \end{aligned} \qquad \text{(MIP)}$$

in which x is a vector of continuous variables and y is a vector of integer variables. The above problem (MIP) specializes to the two following cases:

I. *Mixed-integer linear programming (MILP).* The objective function f, and the constraints h and g are linear in x and y in this case. Furthermore, most of the applications of interest are restricted to the case when the integer variables y are binary, i.e. $y \in \{0,1\}^m$. A number of important classes of problems include the pure integer linear programming problem (only integer variables) and a large number of specialized combinatorial optimization problems that include for instance assignment, knapsack, matching, covering, facility location, networks with fixed charges and traveling salesman problems (see Nemhauser and Wolsey, 1988).

II. *Mixed integer nonlinear programming (MINLP).* The objective function and/or constraints are nonlinear in this case. The most common form is linear in the integer variables and nonlinear in the continuous variables (Grossmann, 1990). More specialized forms include polynomial 0-1 programs and 0-1 multilinear programs which can be transformed into MILP problems (eg see Balas and Mazzola, 1984).

The difficulty that arises in the solution of MILP and MINLP problems is that due to the combinatorial nature of these problems, there are no optimality conditions like in the

continuous case that can be directly exploited for developing efficient solution methods (see paper by Westerberg at this meeting).

In this paper we will concentrate on the modelling and solution of unstructured MHJP problems, and MINLP problems that are linear in the 0-1 variables. Both types of problems correspond to the more general type of mixed-integer optimization problems that arise in batch processing. It is very important however, to recognize that if the model has a more specialized structure, general purpose techniques will be inefficient for solving large scale version of these problems, and specialized combinational optimization algorithms should be used in this case.

Mixed-integer Linear Programming (MILP)

We will assume the more common case in which the subset of the integer variables y are restricted to take only 0 or 1 values. This then gives rise to the MILP problem:

$$\begin{aligned} \min Z = \quad & c^T x + b^T y \qquad \text{(MILP)} \\ \text{s.t.} \quad & Ax + By \wedge d \\ & x \geq 0, \; y \in \{0,1\}^m \end{aligned}$$

In attempting to develop a solution method to solve problem (MILP), the first obvious alternative would be to solve for every combination of 0-1 variables the corresponding LP problem in terms of the variables x, and then pick as the solution the 0-1 combination with lowest objective function. The major drawback with such an approach is that the number of 0-1 combinations is exponential. For example, an MILP problem with 10 0-1 variables would require the solution of $2^{10} = 1024$ LPs, while a problem with 50 0-1 variables would require the solution at $2^{50} = 1.13 \times 10^{15}$ LPs! Thus, this approach is, in general, computationally infeasible.

A second alternative is to relax the 0-1 requirements and treat the variables y as continuous with bounds, $0 \leq y \leq 1$. The problem with such an approach, however, is that except for few special cases (e.g. assignment problem), there is no guarantee that the variables y will take integer values at the relaxed LP solution. As an example, consider the pure integer progra

$$\begin{aligned} \min Z = & -1.2y_1 - y_2 \\ \text{s.t.} \quad & 1.5y_1 - 0.5y_2 \leq 1 \qquad (1) \\ & y_1 + y_2 \leq 1 \\ & y_1, y_2 = 0,1 \end{aligned}$$

By relaxing yi and y2 to be continuous the solution yields the noninteger point yi=0,715, y2=0.285, Z= -1.143. Assume we simply round the variables to the nearest integer value, namely yi = 1, y2 = 0. This, however, is an infeasible solution as it violates the first constraint. In fact, the optimal solution is yi = 0, y2 = 1, Z = -1. Thus, solving the MDLP problem by relaxation of the y variables and rounding them to the nearest integer, will in general not lead to the correct solution. Note, however, that the relaxed LP has the property that its optimal objective value provides a lower bound to the integer solution.

In order to obtain a rigorous solution to the problem (MILP) the most common approach is the branch and bound method which originally was proposed by Land and Doig (1960) and later formalized by Dakin (1965). In the branch and bound technique the objective is to perform an enumeration without having to examine all the 0-1 combinations. The basic idea is first to represent all the 0-1 combinations through a binary tree such as the example shown in Fig. 1. Here at each node of the tree the solution of the linear program subject to integer constraints for the subset of the y variables that are fixed in previous branches is considered. For example, in node A the root of the tree involves the solution of the relaxed LP, while node B involves the solution of the LP with fixed yi = 0, y2 = 1 and withO £y3 ^ 1.

In order to avoid the enumeration of all the nodes in the binary tree, we can exploit the following basic properties. Let k denote a descendent node of node *I* in the tree (e.g. k=B, ^=A) and let (P^) and (P^) denote the corresponding LP subproblems. Then the following properties can be easily established:

1. If (P^) is infeasible then (P^k) is also infeasible.

2. If (P^k) is feasible then (P^) is also feasible, and $(Z')^* \leq (Z^k)^*$. That is, the optimal objective of subproblem (P^) corresponds to a lower bound of the optimal objective at subproblem (P^k).

3. If the optimal solution of subproblem (P^k) is such that y = 0 or 1, then $(Z^k)^* \geq Z^*$. That is, the optimal objective of subproblem (P^k) corresponds to an upper bound of Z*, the optimal MILP solution.

The above properties can be used to fathom nodes in the tree within an enumeration procedure. The question of how to actually enumerate the tree involves the use of branching rules. Firstly, one does not necessarily have to follow the order of the index of the variables y for branching as might be implied in Fig. 1. A simple alternative is to branch instead on the 0-1 variable that is closest to 0.5. Alternatively, one can specify a

priority for the 0-1 variables, or else use a more sophisticated scheme that is based on the use of penalties (Driebeck, 1966; Tomlin, 1971), Secondly, one has to decide as to what node should be examined next having solved the LP at a given node in the tree. Here the two major alternatives are to use a depth-first (last in-first out) or a breadth-first (best second rule) enumeration. In the former case one of the branches of the most recent node is expanded first; if all of them have been examined we backtrack to another node. In the

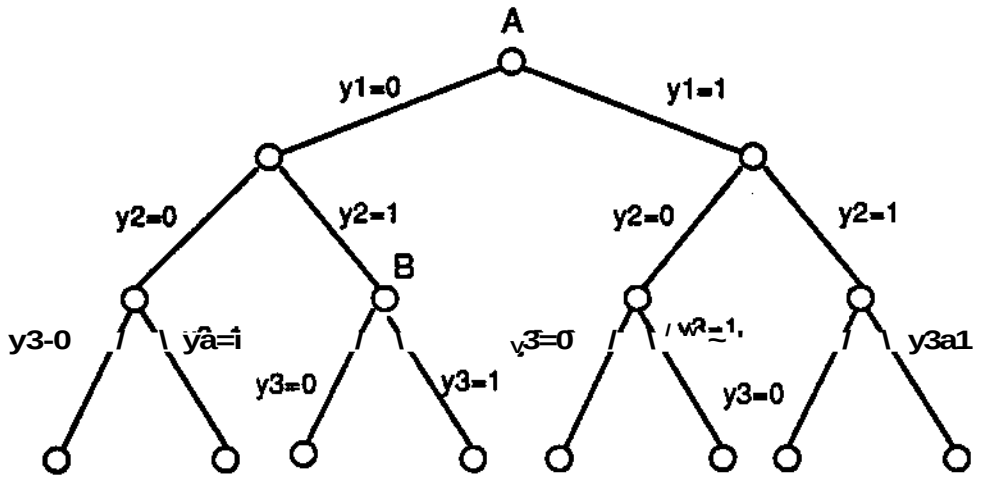


Fig. 1. Binary tree representation **for three 0-1** variables

latter case the two branches of the node with the lowest bound are expanded successively; in this case no backtracking is required. While the depth-first enumeration requires less storage, the breadth-first enumeration requires in general an examination of fewer nodes. In practice the most common scheme is to use depth first, but by branching on the 0 and 1 values of a binary variable at each node.

In summary, the branch and bound method consists in first solving the relaxed LP problem. If y takes integer values we stop. Otherwise we proceed to enumerate the nodes in the tree according to some specified branching rules. At each node the corresponding LP subproblem is solved, typically by updating the dual LP problem of the previous node which requires few pivot operations. By then making use of the properties cited before, we either fathom the node (if infeasible or if lower bound $\geq$ upper bound) or keep it open for further examination. Clearly the computational efficiency is largely dependent on the quality of the lower bounds of the LP subproblems.

As an example, consider the following MILP problem involving one continuous variable and three 0-1 variables:

$$\begin{aligned} \min Z &= x + y_1 + 3y_2 + 2y_3 \\ \text{s.t.} \quad & -x + 3y_1 + 2y_2 + y_3 \geq 0 \\ & -5y_1 - 8y_2 - 3y_3 \leq -9 \\ & x \geq 0,\ y_1, y_2, y_3 = 0,1 \end{aligned} \tag{2}$$

The branch and bound tree using a breadth-first enumeration is shown in Fig. 2. **The number** in **the** circles represents the order in which 9 nodes out of the 15 nodes in the **tree are** examined to find the optimum. Note that the relaxed solution (node 1) has a lower **bound of Z** = 5.8, and that the optimum is found in node 9 where Z = 8, yi=0, y2=y3=l, **and** x=3.

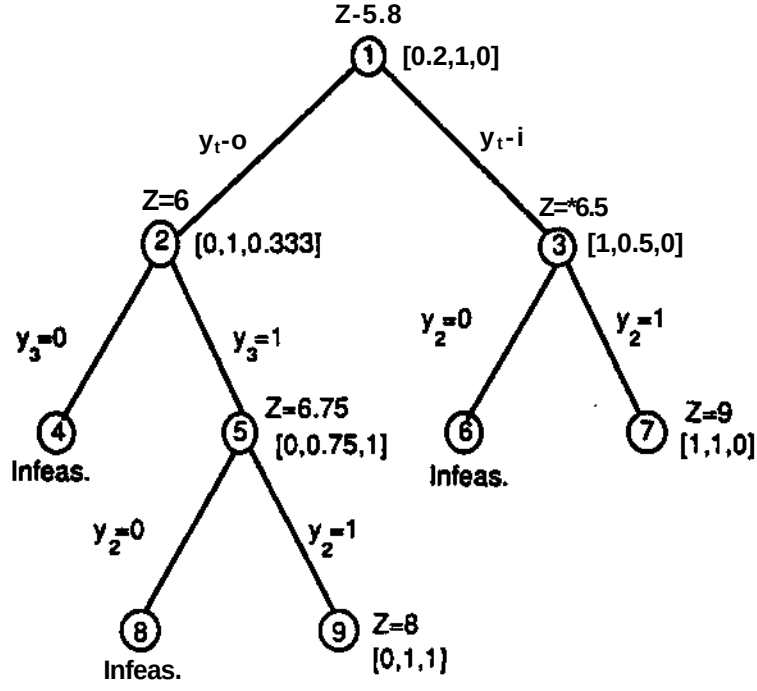


Fig. *2.* Branch and bound tree for example problem (2)

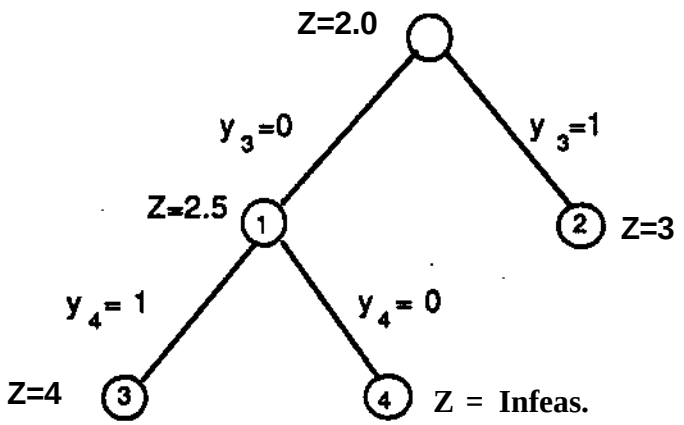
The branch and bound method is currently the most common method used for **MILP** in both academic and commercial computer software (eg. LINDO, ZOOM, **SCICONIC,** OSL, CPLEX, XA). Some of these codes feature a number of special features that can help to reduce the enumeration in the tree search. Perhaps one of the most noteworthy are the generalized upper bound constraints (Beale and Tomlin, 1970) which are integer constraints of the form,

$$\sum_{i \in I} y_i = 1 \tag{3}$$

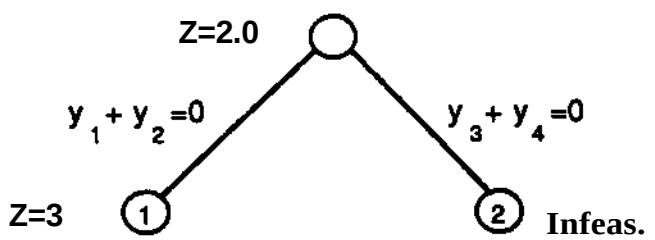
In this case instead of performing branching on individual variables, the branching is performed by partitioning the variables into two subsets (commonly of equal size). As a simple example consider the problem:

$$\begin{aligned} &\text{min. } Z = yi + 2y2 + 3y3 + 4y4 \\ &\text{s.t. } yi + y2 - y3 - y4 \hat{} 0 \\ &\quad yi + Y2 + y3 + y4 = l \\ &yi - 0,1 \quad i - 1,4 \end{aligned} \tag{4}$$

The relaxed UP solution of this problem is Z = 2, yi = *y$* = 0.5, y2 = y4 = 0. If a standard branch and bound search is performed, 4 nodes are required for the enumeration as shown in Fig. 3a. However, if we instead treat the last constraint as a generalized upper bound constraint, only two nodes are enumerated as shown in Fig. 3b.



(a) Branching on individual variables



(b) Branching with generalized upper bounds

Fig.3. Standard branching rule and generalized upper bounds.

Closely related to the generalized upper bound constraints, are the special ordered sets (see Beale and Tomlin, 1970; Tomlin, 1988). The most common are the SOS1 constraints that have the form,

$$\sum_{i \in I} y_i = 1, \qquad x = \sum_{i \in I} a_i y_i \tag{5}$$

in which the second constraint is denoted as a reference row where x is a variable and oci are constants with increasing value. In this case the partitioning of the 0-1 variables at each node is performed according to the placement of the value of the continuous variable x relative to the points (Xi. S0S2 constraints are those in which exactly two adjacent 0-1 variables must be nonzero, and they are commonly used to model piecewise linear concave functions. Again, considerable reductions in the enumeration can be achieved with these types of constraints.

Another important capability in branch and bound codes are preprocessing techniques that have the effect of fixing variables, eliminating redundant constraints, adding logical inequalities, tightening variable bounds and/or performing coefficient reduction (see Brearly et al, 1975; Crowder et al, 1983; Martin and Schrage, 1985). A simple example of coefficient reduction is for instance converting the inequality 2yi + y2 S≥ 1 into yi + y2 ^ 1 which yields a tighter representation in the 0-1 polytope. An example of a logical constraint are minimum cover constraints. For instance, given the constraint 3yi + 2y2 + 4y3 £ 6, yi + y3 ^ 1 is a minimum cover since it eliminates the simultaneous selection of yi = y3 = 1 which violates this constraint. Preprocessing techniques can often reduce the integrality gap of an MILP although their application is not always guaranteed to reduce the computation time.

Although the LP based branch and bound method is the dominant method for MILP optimization, there are other solution approaches which often complement this method. These can be broadly classified into three major types: cutting plane methods, decomposition methods and logic based methods. Only a very brief overview will be given for these methods.

The basic idea of the original cutting plane methods was to solve a sequence of successively tighter linear programming problems. These are obtained by generating additional inequalities that cut-off the fractional integer solution. Gomory (1960) developed a method for generating these cutting planes, but the computational performance tends to be poor due to slow convergence and the large increase in size of the LP subproblems. An alternative approach is to generate strong cutting planes that correspond

to facets, or faces of the integer or mixed-integer convex hull. Strong cutting planes are obtained by considering a separation problem to determine the strongest valid inequality that cuts off the fractional solution. This, however, is computationally a difficult problem (it is NP-hard) and for this reason unless one can obtain theoretically these cuts for problems with special structure, only approximate cutting planes are generated. Also, strong cutting planes are generated from the LP relaxation and during the branch and bound search to tighten the LP. Crowder et al (1983) developed strong cutting planes for pure integer programming problems by considering each constraint individually and treating each of them as knapsack problems. Van Roy and Wolsey (1987) considered special network structures for MILP problems to generate strong cutting planes. In both cases, substantial improvements were obtained in a number of test problems.

A more recent approach for cutting plane methods has been based on the important theoretical result that is possible to transform an unstructured MILP problem into an equivalent LP problem that corresponds to the convex hull of the MILP. This involves converting the MILP into a nonlinear polynomial mixed integer problem which is subsequently linearized through variable transformations (Lovacz and Schrijver, 1989; Sherali and Adams, 1989). Unfortunately, the transformation to the LP with the convex hull is exponential. However, these transformations can be used as a basis for generating cutting planes within a "branch and cut[11] enumeration, and this is for instance an approach that is being explored by Balas et al (1991).

As for decomposition methods for MILP, the most common method is Benders decomposition (Benders, 1962). This method is based on the idea of partitioning the variables into complicating variables (commonly integer variables in the MILP) and noncomplicating variables (continuous variables in MILP). The idea is to solve a sequence of LP subproblems for fixed complicating variables yk,

$$Z^k = \min c^T x + b^T y^k \qquad \text{(LPB)}$$
$$\text{s.t. } Ax \leq d - By^k$$
$$x \geq 0$$

and master problems that correspond to projections in the space of the binary variables and that are based on dual representations of the continuous space. The form of the master problem given K feasible and M infeasible solution points for the subproblems is given by:

$$Z_L^k = \text{mince}$$
$$\alpha \geq (\lambda^k)^T (d - By^k) \quad k = 1, \ldots K \qquad (MB)$$
$$(\mu^m)^T (d - By^k) \leq 0 \quad m = 1, \ldots M$$
$$a \in R^1, \; y \in \{0,1\}^m$$

Since the master problem provides valid lower bounds and the LP subproblems upper bounds, the sequence of problems is solved until equality of the bounds is achieved. Benders decomposition has been successfully applied in some problems (eg. see Geoffrion and Graves, 1976), but it can also have very slow convergence if the LP relaxation is not tight (see Magnanti and Wong, 1981). Nevertheless, this method is in principle attractive in large multiperiod MILP problems. Finally, another type of decomposition techniques are Lagrangean relaxation methods which are applied when complicating constraints destroy the special structure of a problem.

Logic based methods were developed by taking advantage of the analogy between binary and boolean variables. Balas (1974) developed Disjunctive Programming as an alternate form of representation of mixed-integer programming problems. MILP problems are formulated as linear programs with disjunctions (sets of constraints of which at least one must be true). Balas(1975) characterized the family of all valid cutting planes for a disjunctive program. Using these cuts, disjunctions were re-expressed in terms of binary variables and the resulting mixed-integer problem is solved.

Another class of logic based methods are based on using symbolic inference techniques for the solution of pure integer programming problems. Hooker (1988) demonstrated the analogy between unit resolution and first order cutting planes. Jeroslow and Wang (1990) solved the satisfiability problem using a numerical branch and bound based scheme but solving the nodal problems using unit resolution. An alternate symbolic based branching rule was also proposed by these authors. Motivated by the above ideas, Raman and Grossmann (1991) considered the incorporation of logic in general mixed-integer programming problems in the form of redundant constraints to express with logic propositions the relations among units in superstructures. Here one approach is to convert the logic constraints into inequalities and add them to the MILP. Although this has the effect of reducing the integrality gap, the size of the problem is often greatly increased (Raman and Grossmann, 1992a). Therefore, these authors considered an alternate scheme in which symbolic inference techniques were used on a the set of logical constraints which are expressed in either the disjunctive or conjunctive normal form representations. The idea is to perform symbolic inference at each node during the branch and bound procedure in order to perform branching on the variables so as to fix additional binary variables. Orders

of magnitude reductions have been reported by these authors using this approach (Raman and Grossman, 1992b).

Finally, it should be noted that the more recent computer codes for MILP, such as OSL (IBM, 1992) and MINTO (Savelsbergh et al, 1991) have an "open software architecture" that gives the user considerably more flexibility to control the branch and bound search. For instance, these codes allow the addition of cutting planes and modification of branching rules according to procedures supplied by the user.

Mixed-integer Nonlinear Programming (MINLP)

Although the problem (MIP) given earlier in the paper corresponds to an MINLP problem, for most applications the problem is linear in the 0-1 variables and nonlinear in the continuous variables x; that is,

$$\begin{aligned} \min Z = \; & f(x) + c^T y \\ \text{s.t.} \quad & h(x) = 0 \\ & g(x) + By \leq 0 \\ & x \in R^n, \quad y \in \{0,1\}^m \end{aligned} \qquad \text{(MINLP)}$$

This mixed-integer nonlinear program can in principle also be solved with the branch and bound method presented in the previous section (Gupta and Ravindran, 1985; Naber and Schrage, 1990; Borchers and Mitchell, 1991). The major difference here is that the examination of each node requires the solution of a nonlinear program rather than the solution of an LP. Provided the solution of each NLP subproblem is unique, similar properties as in the case of the MBLP would hold with which the rigorous global solution of the MINLP can be guaranteed.

An important drawback of the branch and bound method for MINLP is that the solution of the NLP subproblems can be expensive since they cannot be readily updated as in the case of the MILP. Therefore, in order to reduce the computational expense involved in solving many NLP subproblems, we can resort to two other methods: Generalized Benders decomposition (Geoffrion, 1972) and Outer-Approximation (Duran and Grossmann, 1986). The basic idea in both methods is to solve an alternating sequence of NLP subproblems and MILP master problems. The NLP subproblems are solved by optimizing the continuous variables x for a given fixed value of y, and their solution yields an upper bound to the optimal solution of (MINLP). The MILP master problems consist of linear approximations that are accumulated as iterations proceed, and they have

the objective of predicting new values of the binary variables y as well as a lower bound on the optimal solution. The alternate sequence of NLP subproblems and MILP master problems is continued up to the point where the predicted lower bound of the MILP master is greater or equal than the best upper bound obtained from the NLP subproblems.

The MILP master problem in Generalized Benders decomposition (assuming feasible NLP subproblems) is given at any iteration K by:

$$Z^K_{GB} = \min \alpha \qquad \text{(MGB)}$$
$$\text{s.t} \quad \alpha \leq f(x^*) + c^T y + (\lambda^k)^T [g(x^k) + By] \qquad k=1,2...K$$
$$\alpha \in R^1 \ , \ y \in \{0,1\}^m$$

where a is the largest Lagrangian approximation obtained from the solution of the K NLP subproblems; x^k and λ^k correspond to the optimal solution and multiplier of the kth NLP subproblem; ZQ^K_B corresponds to the predicted lower bound at iteration K.

In the case of the Outer-Approximation method the MILP master problem is given by:

$$Z^K_{OA} = \min \alpha \qquad \text{(MOA)}$$
$$\text{S.t} \quad \alpha \leq f(x^k) + \nabla f(x^k)^T (x - x^k) + c^T y$$
$$T^k \nabla h(x^k)^T (x - x^k) \leq 0 \qquad k=1,2,...K$$
$$g(x^k) + \nabla g(x^k)^T (x - x^k) + By \leq 0$$
$$\alpha \in R^1 \ , x \in R^n , y \in \{0,1\}^m$$

where a is the largest linear approximation of the objective subject to linear approximations of the feasible region obtained from the solution of the K NLP subproblems. T^k is a diagonal matrix whose entries $t_{jj} = \text{sign}(\lambda_j^k)$, where λ_j^k is the Lagrange multiplier of equation hi at iteration k, and is used to relax the equations in the form of inequalities (Kocis and Grossmann, 1987). This method has been implemented in the computer code DICOPT (Kocis and Grossmann, 1989).

Note that in both master problems the predicted lower bounds, ZQ_B, and ZQA, increase monotonically as iterations K proceed since the linear approximations are refined by accumulating the lagrangian (in MGB) or linearizations (in MOA) of previous iterations. It should be noted also that in both cases rigorous lower bounds, and therefore convergence to the global optimum, can only be ensured when certain convexity conditions hold (see Geoffrion, 1972; Duran and Grossmann, 1986).

In comparing the two methods, it should be noted that the lower bounds predicted by the outer approximation method are always greater than or equal to the lower bounds

predicted by Generalized Benders decomposition. This follows from the fact that the Lagrangian cut in GBD represents a surrogate constraint from the linearization in the OA algorithm (Quesada and Grossmann, 1992a). Hence, the Outer-Approximation method will require the solution of fewer NLP subproblems and MILP master problems (see example 4 later in the paper). On the other hand, the MILP master in Outer-Approximation is more expensive to solve so that Generalized Benders may require less time if the NLP subproblems are inexpensive to solve. As discussed in Sahinidis and Grossmann (1991c), fast convergence with GBD can only be achieved if the NLP relaxation is tight

As a simple example of an MINLP consider the problem:

$$
\begin{aligned}
&\min Z = y_i + 1\text{-}5y_2 + 0.5y_3 + x_j\hat{} \quad + X2\hat{} \\
&\text{s.t.} \quad (x_i - 2)^2 - x_2 \leq 0 \\
&\qquad x_i - 2y_i \geq 0 \\
&\qquad x_1 - x_2 - 4(1 - y_2) \leq 0 \\
&\qquad x_1 - (1 - y_1) \geq 0 \\
&\qquad x_2 - y_2 \geq 0 \\
&\qquad x_1 + x_2 \geq 3y_3 \\
&\qquad y_i + y_2 + y_3 \hat{} 1 \\
&\qquad 0 \leq x_i \leq 4, \quad 0 \leq x_2 \leq 4 \\
&\qquad y_1, y_2, y_3 = 0, 1
\end{aligned}
\tag{6}
$$

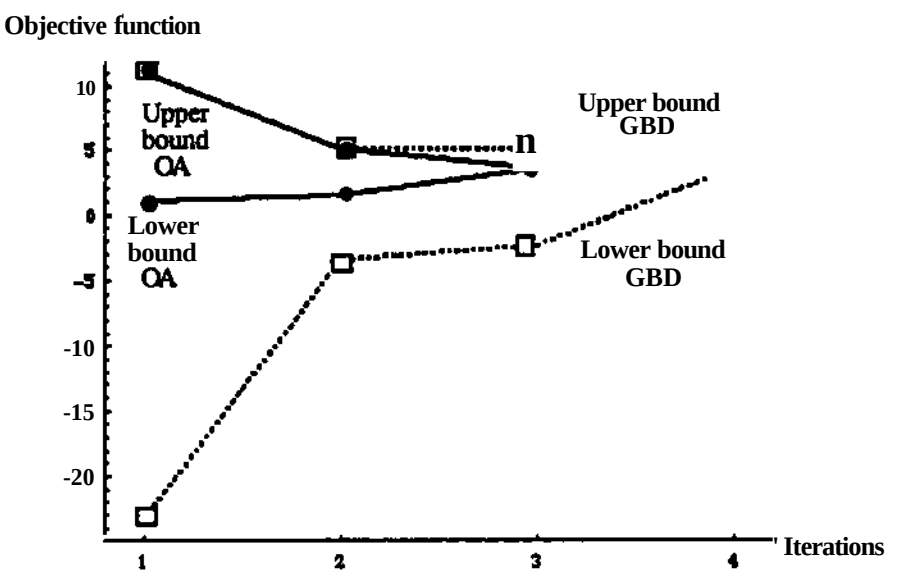


Fig. 4. Progress of iterations of OA and GBD for MINLP in (6).

Note that the nonlinearities involved in problem (6) are convex. Fig. 4 shows the convergence of the OA and the GBD methods to the optimal solution using as a starting point yi as y2 = y3 = 1. The optimal solution is Z =3.5, with $y\backslash$ = 0, y2 = 1, y3 = 0, xi =

1, X2 = 1. Note that the OA algorithm requires 3 major iterations, while GBD requires 4, and that the lower bounds of OA are much stronger.

Other related methods for MINLP include the extension of the OA algorithm by Yuan et al (1989) who considered nonlinear convex terms for the 0-1 variables, and the feasibility technique by Mawekwang and Murtagh (1986) in which a feasible MINLP solution is obtained from the relaxed NLP problem. The latter method has been recently extended by Sugden (1992).

In the application of Generalized Benders decomposition and Outer-Approximation, two major difficulties that can arise are the computational expense involved in the master problem if the number of 0-1 variables is large, and non-convergence to the global optimum due to the nonconvexities involved in the nonlinear functions.

To circumvent the first problem, Quesada and Grossmann (1992a) have proposed for the convex case an LP/NLP based branch and bound method in which the basic idea is to integrate the solution of the MILP master problem and the NLP subproblems which are assumed to be inexpensive to solve. This is accomplished by a tree enumeration in which an NLP is first solved to construct an initial linear approximation to the problem. The LP based branch and bound search is then applied; however when an integer solution is found a new NLP subproblem is solved from which new linear approximations are derived which are then used to update the open nodes in the tree. In this way the cold start for a new branch and bound tree for the MILP master problem is avoided. It should be noted that this computational scheme can be applied to Generalized Benders and Outer-Approximation. As mentioned before the latter will yield stronger lower bounds. However, in this integrated branch and bound method the size of the LP subproblems can potentially become large. To handle this problem Quesada and Grossmann (1992a) proposed the use of partial surrogates that exploit the linear substructures present in an MINLP problem.

In particular, consider that the MINLP has the following structure,

$$
\begin{aligned}
& Z = \min \; c^T y + a^T w + r(v) \\
& \text{st} \quad Cy + Dw + t(v) \leq 0 \qquad \text{(MINLP)} \\
& Ey + Fw + Gv \leq b \\
& y \in Y, \; w \in W, \; v \in V
\end{aligned}
$$

in which the equality constraints are relaxed to inequalities according to the matrix 1* and included in the inequality set. Here the continuous variables x have been partitioned in two subsets w and v such that the constraints are divided into linear and nonlinear constraints, and the continuous variables into linear and nonlinear variables. In this representation

$f(x)=a^T w + r(v)$, $B^T = [C/E]^T$, $g(x)= \begin{bmatrix} Dw - t(v) \\ Fw + Gv \end{bmatrix}$, and $X= WxV$. By constructing a partial surrogate constraint involving the linearization of the nonlinear terms in the objective and nonlinear constraints, the modified master problem has the form:

$$
\begin{aligned}
Z_L^K = \min\ & p && \text{(MMOA)} \\
\text{st}\quad & c^T y + a^T w + b - a = 0 \\
& p \geq r(v^k) + (X^k)^T [\, Cy + Dw + t(v^k) \,] - (\lambda^k)^T G (v - v^k) && k=1,\ldots K \\
& Ey + Fw + Gv \leq b \\
& y \in Y,\ w \in W,\ v \in V,\ a \in R^1,\ p \in R^1
\end{aligned}
$$

where X^k and λ^k are the optimal multipliers of the kth NLP subproblem. It can be seen that as opposed to the Benders cuts, the linearizations are defined in the full space of the variables, requiring only the addition of one new constraint for the nonlinear terms. It can be shown that the lower bound Z_L^K predicted by the above master problem is weaker than the one of OA, but stronger than the one by GBD. Computational experience has shown that the predicted lower bounds are in fact not much weaker than the ones by the OA algorithm.

As for the question of nonconvexities, one approach is to modify the definition of the MELP master problem so as to avoid cutting off feasible mixed-integer solutions. Viswanathan and Grossmann (1990) proposed an augmented-penalty version of the MILP master problem for outer-approximation, which has the following form:

$$
\begin{aligned}
Z_{OA}^K = \min\ & \alpha + \sum_{k=1}^{K} (\rho^k)^T (p^k + q^k) && \text{(MOAP)} \\
\text{s.t.}\quad & \left.\begin{aligned} \alpha \geq f(x^k) + \nabla f(x^k)^T (x - x^k) + c^T y \\ T^k \nabla h(x^k)^T (x - x^k) \leq p^k \\ g(x^k) + \nabla g(x^k)^T (x - x^k) + By \leq q^k \end{aligned}\right\} && k=1,2,\ldots K \\
& \alpha \in R^1,\ x \in R^n,\ y \in \{0,1\}^m
\end{aligned}
$$

in which the slacks p^k, q^k, have been added to the function linearizations, and in the objective function with weights ρ^k that are sufficiently large but finite. Since in this case one cannot guarantee a rigorous lower bound, the search is terminated when there is no further improvement in the solution of the NLP subproblem. This method has been implemented in the computer code DICOPT++ which has shown to be successful in a number of applications. It should also be noted that if the original MINLP is convex the above master problem reduces to the original OA algorithm since the slacks will take a value of zero.

An important limitation with the above approach is that it does not address the question whether the NLP subproblems may contain multiple local solutions. Recently there has an important effort to address the global optimization of nonconvex nonlinear programming problems (e.g. see Horst, 1990). The current methods are either stochastic or deterministic in nature. In the former, generally no assumption about the mathematical structure of the problem is made. Simulated annealing is an example of a method that belongs to this category which in fact has been applied to batch process scheduling (Ku and Karimi, 1991; Patel et al, 1992). This method however has the disadvantages that no strict guarantee can be given about global optimality and that its computational expense can be high. Deterministic methods require the problem to have some particular mathematical structure that can be exploited to ensure global optimality.

Floudas and Visweswaran (1990) developed a global optimization algorithm for the solution of bilinear programming problems. Valid lower and upper bounds on the global optimal solution are obtained through the solution of primal and relaxed dual problems. The primal problem arises by fixing a subset of complicating variables which reduces the bilinear NLP into an LP subproblem. The relaxed dual problems arise from the master problem of GBD but in which the Lagrangian function is linearized and partitioned into subregions to guarantee valid lower bounds. An implicit partition of the feasible space is conducted to reduce the gap between the lower and upper bounds. A potential limitation of this method is that the number of relaxed dual problems to be solved at each iteration can grow exponentially with the number of variables involved in the nonconvex terms.

Another approach for solving nonconvex NLP problems in which the objective function involves bilinear terms is the one presented by Al-Khayyal and Falk (1983). These authors make use of the convex envelopes for the individual bilinear terms to generate a valid lower bound on the global solution. An LP underestimator problem is imbedded in a spatial branch and bound algorithm to find the global optimum. Sherali and Alameddine (1990) presented a reformulation-linearization technique which generates tight LP underestimator problems that dominate the ones of Al-Khayyal and Falk. A similar branch and bound search is conducted to find the global solution. Although this method requires the enumeration of few nodes in the branch and bound tree, it has the main disadvantage that the size of the LP underestimator problems grows exponentially with the number of constraints.

Swaney (1990) has addressed the problem in which the objective function and constraints are given by bilinear terms and separable concave functions. A comprehensive LP underestimator problem provides valid lower bounds that are used within a branch a

bound enumeration scheme in which the partitions do not increase exponentially with the number of variables.

Quesada and Grossmann (1992b) have considered the global optimization of nonconvex NLP problems in which the feasible region is convex and the objective involves rational and/or bilinear terms in addition to convex functions. The basic idea is based on deriving an NLP underestimator problem that involves both linear and nonlinear estimator functions that provide an exact approximation of the boundary of the feasible region. The linear understimators are similar to the ones by Al-Khayyal and Falk (1983), but these are strengthened in the NLP by nonlinear convex underestimators. The NLP underestimator problem, which allows the generation of tight lower bounds of the original problem, is coupled with a spatial branch an bound search procedure for finding the global optimum solution.

Modelling and reformulation

One of the difficulties involved in the application of mixed-integer programming techniques is that problem formulation is not always trivial, and that the way one formulates a problem can have a very large impact in the computational efficiency for the solution. In fact, it is not uncommon that for a given problem one formulation may be essentially unsolvable, while another formulation may make the problem much easier to solve. Thus, model formulation is a crucial step in the application of mixed-integer programming techniques.

While model formulation still remains largely an art, a number of guiding principles are starting to emerge that are based on a better understanding of polyhedral theory in integer programming (see Nemhauser and Wolsey, 1988). In this section we will present an overview of modelling techniques and illustrate them with example problems. These techniques can be broadly classified into logic based methods, semi-heuristic guidelines, reformulation techniques and linearization techniques.

It is often the case in mixed-integer programming that it is not obvious how to formulate a constraint in the first place, let alone formulate the best form of that constraint. Here the use of prepositional logic and its systematic transformation to inequalities with 0-1 variables can be of great help (e.g. see Williams, 1988; Cavalier and Soyster, 1987; Raman and Grossmann, 1991). In particular, when logic expressions are converted into the conjunctive normal form, each clause has the form,

$$P_1 \vee P_2 \vee \ldots \vee P_m \qquad (7)$$

where Pi is a proposition and v is the logical operator OR. The above clause can be readily transformed into an inequality by relating a binary variable yi to the truth value of each proposition Pi (or 1 - yi for its negation). The form of the inequality for the above clause is,

$$yi + yi + \ldots + ym \wedge i \qquad (8)$$

As an example consider the logical condition, Pi v P2 =* P3 which when converted into conjuctive normal form yields, (-1 Pi v P3) A (-1 P2 v P3) . Each of the two clauses can then be translated into the inequalities,

$$1 - yi + y_3 \geq 1 \qquad y_3 \leq yi \qquad (9)$$
$$1 - y_2 + y_3 \wedge 1 \qquad \text{or} \qquad y_3 \geq y_2$$

Similar procedures can be applied when deriving mixed-integer constraints.

Once constraints have been formulated for a given problem, the question that arises is whether alternative formulations might be better suited for the computation, and here the first techniques to be considered are semi-heuristic guidelines. These are rules of thumb on how to formulate "good[11] models. A simple example are variable upper bound constraints for problems with fixed charges,

$$x_i - Uy_i \leq 0 \qquad (10)$$

Here it is well known that although for representation purposes the upper bound U can be large, one should try to select the smallest valid bound in order to avoid a poor LP relaxation. Another well known example is the constraint that often arises in multiperiod MDLP problems (selecting unit z implies possible operation yj in period i, i=l,2,...n),

$$\sum_{i=1}^{n} y_i - n\,z \leq 0 \qquad (11)$$

Here the disaggregation into the constraints,

$$yi - z \leq 0 \quad i=1,2,..n \qquad (12)$$

will produce a much tighter representation (in fact the convex hull of the 0-1 polytope). These are incidentally the constraints that would be obtained if the logical conditions for this constraint are expressed in conjuctive normal form.

The main problem with the disaggregated form of constraints is the potentially large number of them when compared with the aggregated form. Therefore, one has to balance the trade-offs between model size and tighter relaxations- For example Voudouris and Grossmann (1992) report a model in which the disaggregated variable upper bound constraints were used in the form

$$W_{ijsn} \leq U_{ysn}\, y_{jsn} \qquad \forall\, i, j, S, n \qquad (13)$$

An equivalent aggregated form of the above constraints is

$$\sum_{i} W_{ij_{sn}} \leq \sum_{i} U_{ij_{S}n}\, y_{jsn} \qquad \forall\, j, S, n \qquad Q4)$$

When the first set of constraints is used the model involved 708 constraints and required 233 CPUsec using SCICONIC on a VAX 6320. When the second set of constraints is used, the model required only 220 constraints, but the time was increased to 649 sec because of the looser relaxation of (14).

While the above modelling schemes are somewhat obvious, there are some which arc not. A very good example arises in the MILP scheduling model of Kondili et al (1988). In this model, the following constaints apply:

(a) At any time t, an idle item of equipment j can only start at most one task i.ij.

(b) If the item j does start performing a given task i € Ij, then it cannot start any other task until the current one is finished after pi time units.

Kondili et al (1988) formulated the two above constraints as:

$$\sum_{i \in I_j} W_{ijt} \leq 1 \qquad \forall\, j, t$$

$$\sum_{t'=t+1}^{t+p_i-1} W_{ijt'} \leq M(1-W_{ijt}) \qquad \forall\, i,j,t \qquad (15)$$

where M is a suitably large number. Note that the second constraint has the effect of imposing condition (b) if Wyt = 1. As one might expect the second constraint yields a poor relaxation due to the effect of the "big M". Interestingly, Shah et al (1991) found an equivalent representation for the above two constraints, which is not only much tighter, but requires much fewer constraints! These are given by,

$$I \quad \overset{t-pi+1}{I} \quad W_{ijt} * l \quad Vj,t \qquad (16)$$

Thus, this example clearly shows that formulation of a "proper" model is not always trivial or even well understood. Nevertheless, an analysis of the problem based on polyhedral theory can help one understand the reason for the effectiveness of the constraint A detailed proof for constraint (16) is given in the Appendix.

However, not everything in MHJP modelling is an art A more rational approach that has been emerging is the idea of reformulation techniques that are based on variable disaggregation (e.g. see Radrin and Choe, 1979; Jeroslow and Lowe, 1984, 1985), and which have the effect of tightening the LP relaxation. The example *par excellence* is the lot sizing problem that in its "naive[11] form is given by the MHJP (see Nemhauser, Wolsey, 1988):

$$\begin{aligned} & \min \sum_{t=i}^{NT} (Pt*t^{+h}t\ {}^{s}t + {}^{c}tYt) \\ \text{s.t.} \quad & s_t.i + x_t = d_t + s_t \qquad t = l, NT \\ & x_t \leq x^u y_t \qquad t = l, NT \\ & s_o = O \\ & s_t, x_t \wedge 0, \quad y_t \in \{0,1\} \qquad t = l, NT \end{aligned} \qquad (17)$$

where xt is the amount to be produced in period t, yt is the associated 0-1 variable, and st is the the inventory for period t; Ctj^ht, are the set-up, production and storage costs for time period t, t= 1, NT.

As has been shown by Krarup and Bilde (1977) the above MILP can be reformulated by disaggregating the production variables xt into the variables $q_t x$, to represent the amount produced in period t to satisfy the demand in period x ≥. t; that is,

$$Xt = \sum^{NT} q \ll \qquad (18)$$

The MILP is then reformulated as,

$$\begin{aligned} \min \quad & \sum_{t=l}^{NT} \sum_{T=t}^{NT} <Pt^{+h_{t}+h_{t}}+i^{+}-^{+h}*-i)\ Q''^{+} \sum_{t=l}^{NT} {}^{c}tyt \qquad (19) \\ \text{s.t.} \quad & \sum_{t=i}^{C} <ta = dt \qquad t = l, NT \\ & q_{tx} \pounds dty_t \qquad t=l,NT, \quad x = t,NT \\ & q_{tt} \geq 0, \ y_t = \{0,1\} \end{aligned}$$

As it turns out this reformulation yields the absolute tightest LP relaxation since it yields 0-1 values for the y variables; thus this problem can be solved as an LP and there is no need to apply a branch and bound search as there is in the original MDLP (17). It should be noted that although this example is quite impressive, it is not totally surprising from a theoretical viewpoint The lot sizing problem is solvable in polynomial time and therefore one would expect that it should be possible to formulate this problem as an LP that is polynomially sized in the number of variables and constraints. It should be noted that the lot sizing problem is often embedded in MELP planning and scheduling problems with which one can reformulate these problems to tighten the LP relaxation as discussed in Sahinidis and Grossmann (1991b).

Finally, another case that often arises in the modelling of mixed-integer problems are nonlinearities such as bilinear products of 0-1 variables or products of 0-1 with continuous variables. Nonlinearities in binary variables usually involve the transformation of a nonlinear function into a polynomial function of 0-1 variables and then transforming the polynomial function into a linear function of 0-1 variables (Sherali and Adams, 1988). For cross products between binary and continuous variables Petersen (1971) proposed a linearization method which was later extended by Glover (1975). The main idea behind these linearization schemes was the introduction of a new continuous variable to represent the cross product. The equivalence between the bilinear term and the new variable was enforced by introducing a set of equivalence constraints. For the specific case in which the model has a multiple choice structure, an efficient linearization scheme was proposed by Grossmann et al (1992). This scheme compared to the one proposed by Glover gives tighter LP relaxations with fewer number of constraints. The multiple choice structure usually arises in discrete design models, in which the design variables instead of being continuous, take values from a finite set Batch process design problems often involve discrete sizes and as such the latter linearization scheme is well suited. As an example, consider the the bilinear constraints:

$$\sum_{s=1}^{N(i)} d_{is} Y_{is} V_j - P_{ij} W_j \leq 0 \qquad j \in J(i), \quad i=1,..n \tag{20}$$

in which yis is a 0-1 variable and vj is continuous, and where the following constraint holds:

$$\sum_{s=1}^{NO)} y_{is} = 1 \tag{21}$$

In order to remove the bilinear terms $y_{is}v_j$ in (20), define the continuous variables vij_s such that

$$v_j = \sum_{s=1}^{N(i)} v_{ijs} \qquad j \in J(i)\ ,\ i=1,..n \tag{22}$$

$$V_J^L\ y_{is} \le v_{ijs} \le V_J^U\ y_{is} \qquad j \in J(i),\ s=1, N(i),\ i=1..n \tag{23}$$

where V_j^L, V_j^u are valid lower and upper bounds. Using the equations in (21) to (23), the constraints in (20) can be replaced by the linear inequalities

$$\sum_{s=1}^{N(i)} a_{is} v_{ijs} - p_{wj} \le 0 \qquad j \in J(i)\ ,\ i=1,..n \tag{24}$$

The bilinear constraints in (20) can also be linearized by considering in addition to the inequalities in (24), the following constraints proposed by Glover (1975):

$$\begin{aligned} & V_J^L\ y_{is} \le v_{ijs} \le V_J^U\ y_{is} \\ & v_{ijs} \le V_J - V_J^U(1 - y_{is}) \qquad j \in J(i),\ s=1, N(i),\ i=1..n \\ & v_{ijs} \le v_j - v_j^L(1 - y_{is}) \end{aligned} \tag{25}$$

This linearization, however, requires almost twice as many constraints as (22), (23) and (24) Furthermore, while a point (vij_s, vj, yi_s) satisfying (22) and (23) satisfies the inequalities in (25), the converse may not be true. For instance, assume a non-integer point yis such that $vijs = vj^u yis$. Using (21) it follows from (25) that

$$v_j^L\ y_{is} \le V_J \le v_j^U\ y_{is} \tag{26}$$

while (22) yields $vj = vjU$. Thus, the inequalities in (25) may produce a weaker LP relaxation.

For the case when the bilinear constraints in (20) are only inequalities, Torres (1991) has shown that it is sufficient to consider the following constraints from (25):

$$\begin{aligned} & v_j^L\ y_{is} \le v_{ijs} \\ & v_{ijs} \ge v_j - v_j^U(1 - y_{is}) \qquad j \in J(i),\ s=1, N(i),\ i=1..n \end{aligned} \tag{27}$$

which requires fewer constraints than the proposed linearization in (22) and (23). However, the above inequalities also can produce a weaker LP relaxation. For instance, setting $vijs = vjLyis$ for a non-integer point yis yields,

$$V_J \le v_j^1 - (v_j^U - v_j^L)\ y_{is} \tag{28}$$

while (22) yields $vj = vj^L$.

While the modelling techniques described in this section have been mostly aimed at MILP problems, they are of course also applicable to MINLP problems. One aspect however, that is particular to MINLP problems are the modelling of nonlinearities of the continuous variables. In such a case it is important to determine whether the nonlinear constraints are convex or not. If they are not, the first attempt should be to try to convexify the problem. The most common approach is to apply exponential transformations of the form $x = \exp(u)$, where x are the original continuous variables and u the transformed variables; if the original nonlinearities correspond to posynomials these tranformations will lead to convex constraints. A good example is the optimal design of multiproduct plants with single product campaigns (Grossmann and Sargent, 1978), with which Kocis and Grossmann (1988) were able to rigorously solve the MINLP problem to global optimality with the outer-approximation algorithm. When no transformations can be found to convexify a problem, this does not necessarily mean that the relaxed NLP has multiple local optima. However, nonconvexities in the form of bilinear products and rational terms are warning signals that should not be ignored. In this case the application of a global optimization method such as the ones described previously will be the only way to rigorously guarantee the global optimum.

Finally, it should be noted that another aspect in modelling is the computer software that is available for formulating and solving mixed-integer optimization problems. At this point the modelling system GAMS (Brooke et al. 1988) has emerged as a major tool in which problems can be specified in algebraic form and automatically interfaced with codes for mixed-integer linear and nonlinear optimization (e.g. ZOOM, SCICONIC, OSL, DICOPT++). Modelling tools such as this one greatly reduce the time that is required to test and prototype mixed-integer optimization models.

Examples

In this section we will present several examples to illustrate a number of points and the application of techniques for mixed-integer programming in batch processing.

Example 1: The MILP for the State Task Network model for the scheduling of batch operations by Kondili et al. (1991) has been used to compare the performance of three MILP codes: ZOOM an academic code, and OSL and SCICONIC which are commercial codes. This example which has 5 tasks, 4 units, 10 time periods and 9 states (see Fig. 5), also demonstrates the effect of modelling schemes on the solution efficiency. The objective is to maximize the production of the two final products. The resulting MILP model, which incorporates the constraints in (16), involves 251 variables (80 binary) and

309 constraints (see Shah et a1, 1991). The results of the benchmark comparison between **the three codes for** this problem are shown in Table 1. The problems were solved to optimality (0% gap) by using GAMS as an interface. As can be seen in Table 1 the **performance** of **the** codes is quite different SCICONIC had the lowest computing **requirements: about** less than a tenth of the requirements of ZOOM.

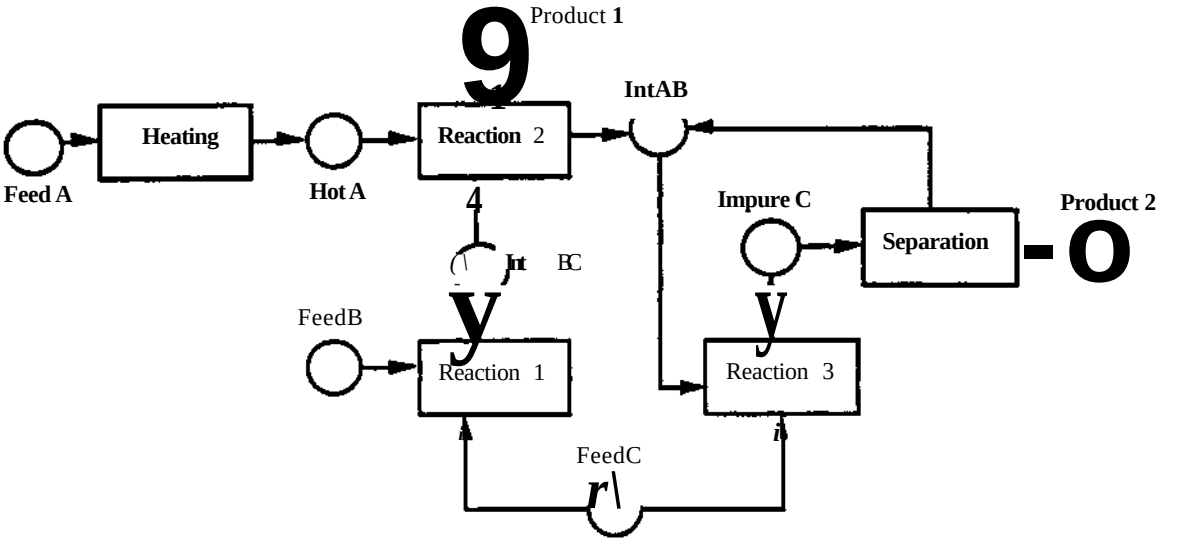


Fig. 5. State-task network for example problem

It should be noted that Shah et al (1991) solved this problem with their own branch and bound method in two forms. In the first the MILP was identical as the one solved in this paper. In this case 1085 nodes and 437 sees on a SUN1- Sparcstation were required to solve the problem within 1% of optimality. In the second form, the authors applied a solution strategy for reducing the size of the relaxed LP and for reducing degeneracies. In this case only 29 nodes and 7 sees were required to solve the problem, which represents a performance comparable to the one by SCICONIC.

To illustrate the effect that alternate formulations may have, two cases were considered and the results are shown in Table 2a. Firstly, realizing that the objective function does not contain any binary variable, the second column involves the addition of a penalty to the objective function in which all the binary variables are multiplied by a very small number so as not to affect the optimum solution. The idea is simply to drive the 0-1 variables to zero to reduce the effect of degeneracies. In the third column, 18 logic cuts in the form of inequalities have been added to the MILP model to reduce the relaxation gap (Raman and Grossmann, 1992a). These logic cuts represent connectivity of units in the state task network. For example, in the problem of Fig. 5, since no storage of impure C is

allowed, the separation step has to be immediately performed after reaction 3. As can be seen from Table 2a, both modelling schemes lead to a substantial improvement in the solution efficiency with OSL, while with SCICONIC only the addition of logic cuts improves the solution efficiency. Furthermore, the effect of adding these logic cuts in this problem have been studied for the case of 10, 20, 40 and 50 time periods. The results, shown in Table 2b, demonstrate an increase in the effectiveness of the logic cuts in improving the efficiency of the branch and bound procedure. The reduction in the number of nodes required in the branch and bound search due to the logic cuts increases from a factor of 3 for the 10 period case to a factor of more than 6 for the 40 period case. The 50 time period problem, with 1251 variables (400 binary) and 1509 constraints could not be solved by OSL within 100,000 iterations and 1 hour of CPU time on the IBM POWER 530. With the addition of the logic cuts, the problem is solved in 158-84 sec requiring only 698 nodes and 5017 iterations.

Table 1. Comparison with several **MILP** codes

	nodes	iterations	CPU time *
ZOOM	410	7866	39.44
OSL	350	918	14.85
SCICONIC	61	318	3.63

*** IBM POWER 530**

Table 2a. Computational results with modified formulations (OSL / SCICONIC)

	Original Model	**Model with altered Objective Function**	**Model with Logic Cuts**
number of nodes	**350/61**	**40/61**	**108/33**
number of iterations	**918/318**	**336/318**	**620/233**
CPU time*	**14.85/3.63**	**2.51/3.65**	**5.98/2.13**
Relaxed Optimum	**257.2**	**257.2**	**257.2**
Integer Optimum	**241**	**241**	**241**

*** sec IBM POWER 530**

Table 2b. Example 1: Effect of logic cuts for different time periods

	Original Model	Model with Logic Cuts
10 Time Periods 251 variables 80 binary		
Constraints	309	327
Number of nodes	350	108
Number of Iterations	918	620
CPU Time *	14.85	5.98
20 Time Periods 501 variables 160 binary		
Constraints	609	643
Number of nodes	123	67
Number of Iterations	755	658
CPU Time *	10.22	7.81
40 Time Periods 1001 variables 320 binary		
Constraints	1209	1279
Number of nodes	2098	315
Number of Iterations	25964	3423
CPU Time •	424.68	67.17
50 Time Periods 1251 variables 400 binary		
Constraints	1509	1597
Number of nodes	>20,000	698
Number of Iterations	>100,000	5017
CPU Time*	>3,600	158.84

*secIBM POWER530

Example 2. In order to illustrate the effect of preprocessing and the use of SOSl constraints, consider the design of multiproduct batch plants with one unit per stage, operating with single product campaigns, and where the equipment is available in discrete sizes (Voudouris and Grossmann, 1992a). The MILP model is as follows:

$$\min \quad \sum_j \sum_s y_{j,s} \qquad \text{(RP1)}$$

$$\text{s.t.} \qquad T_i \geq \sum_s \left(\frac{Q_i S_{ij}}{v_{js}}\right) y_{js} \; T_{Li} \qquad i=1,..,N \;,\; j=1,..,M$$

$$\sum_i T_i \leq H$$

$$\sum_s y_{js} = 1 \qquad J=1,..,M$$

$$T_i \geq 0 \quad i=1,..,N \;, \quad y_{js} \in \{0, 1\} \quad j=1,..,M \;, \; s=1,..,nsj$$

The example considered involves with 6 stages and 5 products. To illustrate the effect that the number of discrete sizes has in the size of model (RPl) as well as in the computational performance, three problems one with 8, one with 15 and another with 29 discrete sizes were considered.The MILP problems were solved using SCICONIC 2.11 (SCICONIC, 1991) through GAMS 2.25 in a Vax-6420.

Table 3: Computational results for example 2

# DIS. SIZES	CONSTRAINTS	VARIABLES	0-1 VAR'S	CPU-TIME •	ITERATIONS	NODES
WITHOUT SOSI. DOMAIN REDUCTION AND CUTOFF						
8	38	54	48	2.93	181	89
15	38	96	90	25.09	985	731
29	38	180	174	44.94	1203	979
WITH SOSI. DOMAIN REDUCTION AND CUTOFF						
8	38	"4&mm	40	1.93	5V	53
15	38	82	76	2.85	91	64
29	38	154	148	6.64	182	150

• IN VAX-6420 SECONDS

As seen from Table 3, the number of discrete sizes has a significant effect in **the number** of **0-1** variables, and hence in the number of iterations and the CPU time. One can, **however, reduce** significantly the computational requirements by performing **a** domain **reduction of the 0-1** variables through the use of bounds to fix a subset of them to zero, **treating the** multiple choice constraints as SOSl constraints and applying and objective function cutoff as described in Voudouris and Grossmann (1992a). As seen in Table 3, reductions of up to one order of magnitude are achieved.

Example 3. This example will illustrate how strong cutting planes may significantly improve the computational performance of MILP problems with poor continuous relaxations. A good example are jobshop scheduling problems. Consider the case in which one has to schedule a total of 8 batches, 2 for each one of 4 products A, B, C, D so as to minimize the makespan in a plant consisting of 5 stages. The processing times for each product are given in Fig. 6, where it can be seen that not all products require all the stages, and that they all require zero-wait transfer policy.

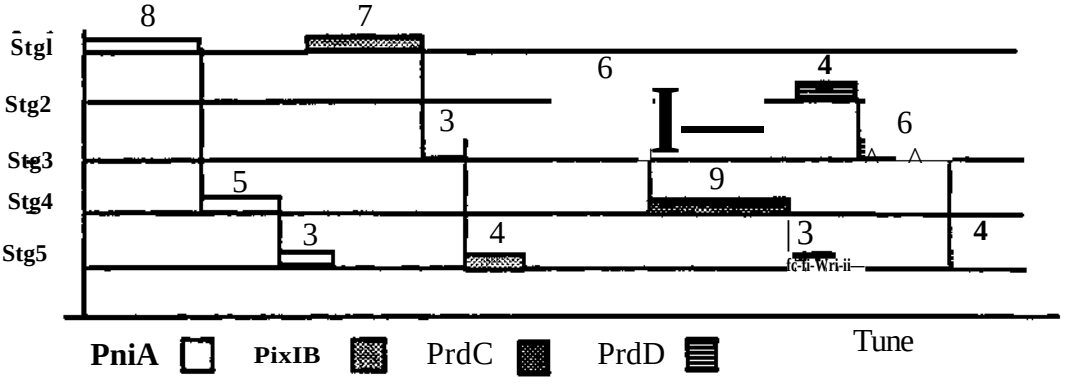


Figure 6. Processing times of products in various stages.

As noted in Voudouris and Grossmann (1992b) the makespan minimization problem described above can be formulated as an MILP problem of the form:

$$\min \quad Ms \qquad \text{(PI)}$$

$$\text{s.t} \quad Ms \geq S_i + \sum_{k=1}^{M} t_{ik} \qquad \forall i$$

$$S_j - S_i + W(1-y_{ijk}) \geq \sum_{k=1}^{k} t_{ik} - \sum_{k=1}^{k-1} t_{jk} \qquad \forall (i,j,k') \in C$$

$$S_i - S_j + W y_{ijk} \geq \sum_{k=1}^{k'} t_{jk} - \sum_{k=1}^{k'-1} t_{ik} \qquad \forall (i,j,k') \in C$$

$$y_{ijk} \in \{0,1\} \qquad \forall i,j,k \qquad S_i \geq 0 \quad \forall i$$

In the above formulation the potential clashes at every stage are resolved with a pair of disjunctive constraints that involve a large bound W. The difficulty with these constraints is **that** they are trivially satisfied when the corresponding binary variables are relaxed, which in turn yields a poor LP relaxation. For this example, the LP relaxation had an objective value of 18 compared to the optimal integer solution of 41, which corresponds to

a relaxation gap of 56%. The MILP was solved with SCICONIC 2.11 on a Vax-6420 requiring 55 CPUsecs, and the solution is shown in Fig.7. In order to improve the LP relaxation basic-cut inequalities have been recently proposed by Applegate and Cook (1991), and they have the form,

$$\sum_{i \in T} t_{ik} S_{ik} \geq E_T k t_{ik} + \sum_{\{i \in T,\, j \in T,\, i<j\}} t_{ik} t_{jk} \qquad Vk$$

$$\sum_{i \in T} t_{ik} (Ms - S_{ik}) \geq Fr_k t_{ik} + \sum_{i \in T} t_{ik}^2 + \sum_{\{i \in T, j \in T, k j\}} t_{ik} t_{jk} \qquad Vk$$

where Si_k = the starting time of job i on machine k ($Si_k = Si + \sum_{k<s<l}^{M} t_{ik}$)

T s a subset of the set of jobs

$E^{\wedge}$ = the earliest possible starting time of j on k (which is just the sum of j's processing times on the machines before k)

$I^{\wedge}$ = the minimum of E_{jk} over all j € T

Fj_k = the minimum completion time of j after it is processed on k (which is just the sum of j^fs processing times on the remaining machines)

F_{Tk} = the minimum of Ej_k over all j € T

The impact of these constraints in the MILP was very significant in this example. The LP relaxation increased to 38 which corresponds to a gap of only 7%. In this case the optimal solution was obtained in only 8 CPUsecs.

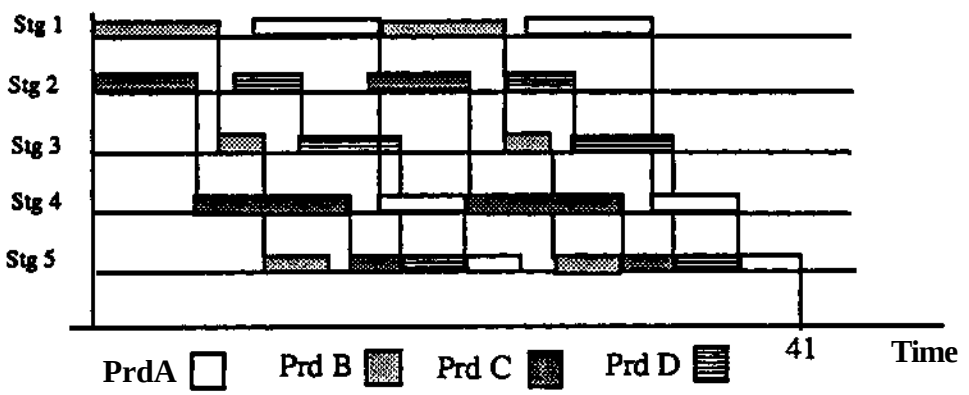


Figure 7. Optimal schedule for example 3.

Example 4. The optimal design of multiproduct batch plants with parallel units operating out of phase (see Fig. 8) will be used to illustrate the computational performance of the different MINLP algorithms.

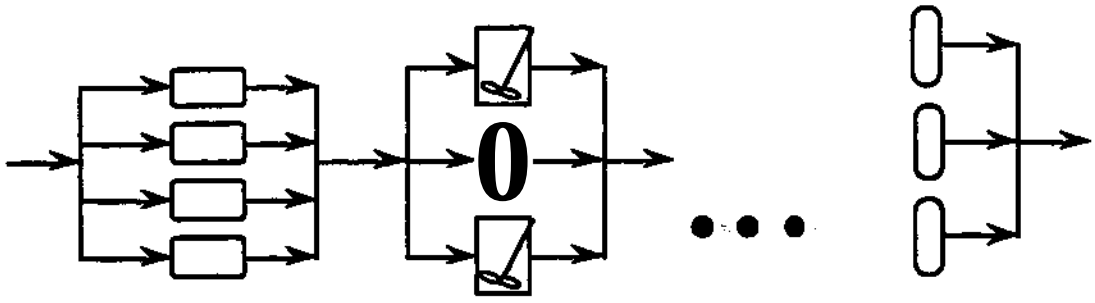


Fig. 8. **Multiproduct** batch **plant** with **parallel** units

Two different cases are considered. One consists of 5 products in a plant with 6 processing stages process with a maximum of 4 parallel units per stage (batch5). The second one has 6 products in a plant with 10 stages and a maximum of 4 parallel units per stage (batch6). The MINLP formulation and the data for these examples are reported in Kocis and Grossmann (1989) and the size of the problems is given in Table 4. The model is a geometric programming problem that can be transformed into a convex MINLP through exponential transformations.

Table 4 : Data for problems in **Example 4.**

problem	Binary variables	Continuous variables	Constraints
batch5	24	22	73
batch6	40	32	141

The GBD and OA algorithms were used for the solution of both examples and the computational results are given in Table 5. The GBD algorithm was implemented within GAMS, while the version of the OA algorithm used was the one implemented in DICOPT++ with the augmented penalty. MINOS 5.2 was used for the NLP subproblems and SCICONIC 2.11 for the MELP master problems. Note that in both cases the OA algorithm required much fewer iterations than GBD which predicted very weak bounds and a large number of infeasible NLP subproblems during the iterations. For problem batch5 both algorithms found the global optimal solution. For batch6, both algorithms also found the same solution which however is suboptimal since the correct optimum is $398,580. In the case GBD, the algorithm did not converge as it had a large gap between the lower and upper bounds after 66 iterations. In the case of the OA algorithm as implemented in DICOPT ++ the optimal solution was suboptimal due to the termination criterion used in this implementation.

Table 5 : Computational results of Example 4

problem	GBD algorithm			OA algorithm		
	Solution	iterations	CPU time*	Solution	iterations	CPU time*
batchS	$285,506	67	766.88	$285,506	3	26.94
batch6	$402,496	66+	2527.2	$402,496	4	108.58

* sec Vax 642U + Convergence ot bounds was not achieved

In both the above examples the solution of the MILP master problem in the OA algorithm was of the order of 80%. A rigorous implementation of the OA algorithm for the convex case (Duran and Grossmann, 1986) and the LP/NLP based branch and bound algorithm by Quesada and Grossmann (1992a) were also applied to compare their computational performance with respect to the number of nodes that are required for the **MILP** master problem. The results are given in Table 6. As can be seen, both algorithms required the solution of 4 and 10 NLP subproblems, respectively, and they both obtained the same optimal solution. However, the LP/NLP based branch and bound required a substantially smaller number of nodes (36% and 16% of the number of nodes required by OA).

Table 6. Results on MILP solution step for problems in Example 4.

problem	optimal solution	Outer Approximation		LP/NLP branch and bound	
		nodes	NLP	nodes	NLP
batch 5	$285, 506	90	**4**	32	4
batch 6	$398, 580	523	**10**	84	10

Example 5. In order to illustrate the effect of nonconvexities, consider the design and production planning of a multiproduct batch plant with one unit per stage. The objective is to maximize the profit given by the income from the sales of the products minus the investment cost. Lower bounds are specified for the demands of the products and the investment cost is assumed to be given by a linear cost function. Since the sizes of the vessels are assumed to be continuous, this gives rise to the following NLP model:

$$\max \quad P = \sum_i p_i n_i B_i - \sum_j \alpha_j V_j$$

$$\text{s.t.} \quad V_j \geq S \; B_i \qquad i=I, N, \; j=1,M \qquad \text{(NLPP)}$$

$$\sum_i n_i Ti \leq H$$

$$\frac{Q_i^L}{n_i} - B_i \leq 0 \qquad i=I, N$$

$$V_j, B_i, n_i \geq 0$$

where iii and Bjis the number and size of the batches for product i, and Vj is the size of the equipment at stage j. The first inequality is the capacity constraint in terms of the size factors Sy, the second is the horizon constraint in terms of the cycle times for each product Ti and the total time H, and the last inequality is the specification on lower bounds for the demands Qi^L. Note that the objective function is nonconvex as it involves bilinear terms, while the constraints are convex. The data for this example are given in Table 7. A maximum size of 5000 L was specified for the units in each stage.

Table 7. Data for Example 5

Product	T_i (hrs)	Pi ($/Kg)	Qi^L (Kg)	Sij(L/kg) 1	2	3
A	16	15	80000	2	3	4
B	12	13	50000	4	6	3
C	13.6	14	50000	3	2	5
D	18.4	17	25000	4	3	4

a_t = 50, oa_2 = 80, a_3 = 60 ($/L); H = 8,000 hrs

When a standard local search algorithm (MINOS 5.2) is used for solving this NLP problem using as a starting point nA=nB=nc=60 and no=300 the predicted optimum profit is $8,043,800/yr and the corresponding batch sizes and their number are shown in Table 8.

Table 8. Suboptimal solution for example 5

	A	B	C	D
B	1250	833.33	1000	1250
n	79.15	60	50	289.868

Since the formulation in (NLPP) is nonconvex there is no guarantee that this solution is the global optimum. This problem can be reformulated by replacing the nonconvex terms by underestimator functions to generate a valid NLP underestimator problem as discussed in Quesada and Grossmann (1992b). The underestimator functions require the solution of LP subproblems to obtain tight bounds on the variables, and yield a convex NLP problem with 8 additional constraints.

The optimal profit predicted by the nonlinear underestimator problem is $8,128,100/yr with the variables given in Table 9. When the objective function of the original problem (NLPP) is evaluated for this feasible point the same value of the objective function is obtained proving that it corresponds to the global optimal solution. This problem was solved on a IBM/R6000-530 with MINOS 5.2, and 1.6 sees were required to solve the LP bounding problems and 0.26 sees to solve the NLP underestimator problem.

It is interesting to note that both the local and global solutions had the maximum equipment sizes. The only difference was in the number of batches produced for products A and D.

Table 9. Global optimum solution for example 5

	A	B	C	D
B	1250	833.33	1000	1250
n	389.5	60	50	20

Concluding Remarks

This paper has given a general overview of mixed-integer optimization techniques for the optimization of batch processing systems. As was shown with the review of previous work, the application of these techniques has increased substantially over the last few years. Also, as was discussed in the review of mixed-integer optimization techniques, a number of new methods are emerging that have the potential of increasing the size and scope of the problems to be solved. While in the case of MILP branch and bound methods continue to play a dominant role, the use of strong cutting planes, reformulation techniques and the integration of symbolic logic hold great promise for reducing the computational expense for solving large scale problems. Also, it will be interesting to see in the future what impact interior point methods will have on MILP optimization (see for instance Borchers, and Mitchell, 1991, for preliminary experience). As was also shown with the results, different computer codes for MILP can show very large differences in performance despite the fact that they all rely on similar ideas. This clearly points to the importance of issues such as software and hardware implementation, preprocessing, numerical stability and branching rules-

In the case of MINLP, the application of this type of models is becoming more widespread with the Outer-Approximation and Generalized Benders Decomposition methods. The former has proved to be generally more efficient, although the latter is better suited for exploiting the structure of problems (e.g. see Sahinidis and Grossmann, 1991a). Aside from the issue of problem size in MINLP optimization, nonconvexities remain a major source of difficulties. However, significant progress is being made in the global optimization of nonconvex NLP problems, and this will surely have a positive effect on MINLP optimization in the future. Finally, as has been emphasized in this paper, problem formulation for MILP and MINLP problems has often a very large impact in the efficiency of the computations, and in many ways still remains an art for the application of these techniques. However, a better

understanding of polyhedral theory and establishing firmer links with symbolic logic may have a substantial effect on how to systematically formulate mixed-integer problems.

Acknowledgment

The authors gratefully acknowledge financial support from the National Science Foundation under Grant CBT-8908735, and from the Engineering Design Research Center at Carnegie Mellon.

Appendix : On the reduced set of inequalities by Shah *et al* (1991).

In order to prove the equivalence of constraint (15) and the one in (16) by Shah *et al* (1991), one must first state the following lemma

Lemma : The integer constraint,

$$y_i+Y_2+...+y_{K+i} \leq 1 \qquad (1)$$

is equivalent to and sharper than the set of integer constraints

$$y_i+y_2+\text{-}+y_{ic} \leq 1 \qquad (AO)$$
$$y_{K+1}+y_i \wedge i \qquad (A1)$$
$$y_{K+1}+y_2 \wedge i \qquad (A2)$$
$$\ldots$$
$$y_{K+1}+y_K \wedge i \qquad (AK)$$

Proof :

First we note that (1) can easily be seen to be equivalent to the constraints (AO) to (AK) since in these at most one variable y can take an integer value of 1. Multiplying constraint (AO) by (K-l) and adding it with the constraints in (Al)-(AK) yields,

$$K(y_i+y_2+..+y_{K+i}) \wedge K-l+K$$

$$y_i+y_2+..+y_{K+i} \wedge 1+(K-1)/K$$

Since yi **e** {0,1}, the right hand side can be rounded below to obtain the inequality

$$y_i+y_2+"+y_{ic}+i \wedge i$$

Proof of constraint :

We know, from (15)

$$\sum_{i(j,t} w_{i(j,t} \leq i \qquad \forall j,t \qquad w_{ijt} \in \{0,1\} \quad \forall ij.t$$

ie. $W_{iij.t}+W_{i2o:t}+..+W_{i»jj.t} \leq 1 \qquad \text{forall} j,t$

If pii > 1, since no unit can process two tasks simultaneously,

$$W_{i1,j,t-1}+W_{i1\,i,t} \leq 1$$
$$W_{i1,j,t-1}+W_{iz.j.t} \leq 1$$
$$..$$
$$W_{i1,j,t-1}+W_{inj.j.t} \leq 1$$

From the lemma, we get

$$W_{i1,j,t-1}+W_{i1,j,t}+W_{i2,j,t}+..+W_{inj,j,t} \leq 1$$

If **Pi2**> 1

$$W_{i2.i.t-1}+W_{i1.i.t} \leq 1$$
$$\leq 1$$
$$\ldots$$
$$W_{i2,j,t-1}+W_{inj,j,t} \leq 1$$

Also, $W_i2,j,t-i+Wii,_j,n£ 1$

This leads to $Wi2,j,t-i^{+}Wii,j.t-i+Wii,_j,t+Wi2.j,_t+..+W_{5n}j,j,t £ 1$

Repeat for all W^t-i where pi[1] > 1 to get

$$Wii,_j,M+Wi2.j.H+..+W_{in}j,_j,_t.i+W_ii,_j,_t+Wi2j,t+"+Wi_{nj},j,_t \leq 1$$

Now, if pa > 2

$$W \quad \cdot W \quad _{t-1} \leq 1$$
$$Wiij,_t.2+Wi2j,H \leq 1$$
$$\ldots$$
$$Wii,j,t-2+W_{inj>j>t}.i \leq 1$$
$$Wuj.rf+Wuj.t \wedge 1$$
$$\_ \quad 1$$
$$W_{il},j.,_t.2+W_{i2}.j.t <$$
$$W_{inj,j,t} \leq 1$$
$$W_{il},j,_t.2+$$

From the lemma, we get

$$W_{i1,j,t-2}+W_{i1,j,t-1}+W_{i2,j,t-1}+..+W_{inj,j,t-1}+W_{i1,j,t}+W_{i2,j,t}+..+W_{inj,j,t} \leq 1$$

Repeat for all $Wr,j,_t$-2 for pr > 2

Repeat for all $Wj'_{,j},_t.3$ **for pr > 3**

...

Repeat for all $Wi'_{,j},_{t-Pi+}i$ **for pr > pi-i**

Finally, we get

$$W_{i1,j,t\ i1+1}+.+W_{i1,j,t-1}+W_{iij,t}+Wi2j,t-\wedge 1+W_{i2,j,t-1}+.+W_{i2,j,t}+W_{inj,j,t-pnj+1}+W_{inj,j,t-1}+.+Wn,jj._t \wedge 1$$

Grouping terms in **the** above inequality yields

$$\sum_{f=t-pl+1}^{t} W_{il.i.f} + \sum_{t\wedge t-pl+1}^{t} W_{i2.j.f} + \cdots + \sum_{t'=t-pnj+1}^{t} W_{inj.j.f} \leq 1$$

Further summing over all i, we get the constraint by Shah et al. (1991)

$$\sum_{ielj} \sum_{t'=t-pi+1}^{t} Wy., < 1$$

References

Al-Khayyal, F.A. and Falk, J.E. (1983) Jointly constrained biconvex programming, *Mathematics of Operations Research* 8,273-286.

Applegate D. and Cook W. (1991). A Computational Study of the Job-Shop Scheduling Problem, *ORSA Journal on Computing,* 3, No. 2, pp 149-156.

Balas, E (1974). "Disjunctive Programming : Properties of the Convex Hull of Feasible Points." MSRR #348, Carnegie Mellon University.

Balas, E. (1975). "Disjunctive Programming : Cutting Planes from Logical Conditions". Nonlinear Programming 2, 0. L. Mangasarian et al., eds., Academic Press, 279-312.

Balas, E., Ceria, S. and Cornuejols, G. (1991). A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs. MSRR No. 576, Carnegie Mellon University.

Balas, E., and Mazzola, J.B. (1984). Nonlinear 0-1 Programming: Linearization Techniques. *Mathematical Programming,* 30,1-21.

Beale, E. M. L, and Tomlin, J. A. (1970). Special Facilities in a Mathematical programming System for Nonconvex problems using Ordered Set of Variables, in Proceedings of the *Fifth International Conference on Operational Research** J. Lawrence, ed., Tavistock Publications, pp 447-454.

Benders, J. F. (1962). Partitioning Procedures for Solving Mixed Integer Variables Programming Problems, *Numerische Mathematik,* 4, 238-252.

Birewar D.B and Grossmann I.E (1990). Simultaneous Synthesis, Sizing and Scheduling of Multiproduct Batch Plants, *Ind. Eng. Chem. Res.,* Vol 29, Nol 1, pp 2242-2251

Borchers, B. and Mitchell, J.E. (1991). Using an Interior Point Method in a Branch and Bound Method for Integer Programming, R.P.I. Math. Report No. 195.

Borchers, B. and Mitchell, J.E. (1991). An Improved Branch and Bound Algorithm for Mixed-Integer Nonlinear Programs, R.P.I. Math. Report No. 200.

Brearly, A.L., Mitra, G. and Williams, H.P. (1975). An Analysis of Mathematical Programming Problems Prior to Applying the Simplex Method, *Mathematical Programming,* 8, 54-83.

Brooke, A., Kendrick, D. and Meeraus, A. (1988). GAMS: A User's Guide. Scientific Press, Palo Alto.

Cavalier, T. M. and Soyster, A. L. (1987). Logical Deduction via Linear Programming. IMSE Working Papa: 87-147, Dept of Industrial and Management Systems Engineering, Pennsyvania State University.

Crowder, H. P., Johnson, E. L., and Padberg, M. W. (1983). Solving Large-Scale Zero-One Linear Programming Problems, *Operations Research,* 31,803-834.

Dakin, R. J. (1965). A Tree search Algorithm for Mixed Integer Programming Problems, *Computer Journal,* 8,250-255.

Driebeek, N., J. (1966). An Algorithm for the solution of Mixed Integer Programming Problems, *Management Science,* 12, 576-587.

Duran, M.A. and Grossmann, I.E. (1986). An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs. *Mathematical Programming* 36,307-339.

Faqir N*M* and Karimi I.A (1990). Design of Multipurpose Batch Plants with Multiple Production Routes, *Proceedings FOCAPD'89,* Snowmass Village CO, pp 451-468

Fletcher R, Hall J.A. and Johns W.R. (1991). Flexible Retrofit Design of Multiproduct Batch Plants, *Comp & Chem. Eng.* 15, 843-852

Floudas, C A. and Visweswaran, V. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs-I Theory, *Computes chem. Engng.* 14,1397-1417

Geoffrion, A.M. (1972). Generalized Benders Decomposition. *Journal of Optimization Theory and Applications,* **10(4),** 237-260.

Geoffrion,A.M. and Graves, G. (1974). Multicommodity Distribution System Design by Benders Decomposition, *Management Science,* 20, 822-844.

Glover F.(1975). Improved Linear Integer Programming Formulations of Nonlinear Integer Problems, *Management Science,* Vol. 22, No. 4, pp 455-460

Gomory, R. E. (1960). An Algorithm for the Mixed Integer Problem, RM-2597, The Rand Corporation..

Grossmann, I.E. (1990). Mixed-Integer Nonlinear Programming Techniques for the Synthesis of Engineering Systems, *Research in Eng. Design,* 1,205-228.

Grossmann LE and Sargent R.W.H, (1979) Optimum Design of Multipurpose Chemical Plants , *Ind£ng.Chem.ProcJDesDev.,* Vol 18, No. 2, pp 343-348

Grossmann *IE,* Voudouris V.T., Ghattas 0.(1992). Mixed-Integer Linear Programming Reformulation for Some Nonlinear Discrete Design Optimization Problems, Recent Advances in Global Optimization (eds. Floudas, C.A.. and Pardalos, P.M.), pp.478-512, Princeton University Press

Gupta, J.N.D. (1976). Optimal Flowshop Schedules with no Intermediate Storage Space. *Naval Res. Logis. Q.* 23, 235-243.

Gupta, O.K. and Ravindran, V. (1985). Branch and Bound Experiments in Convex Nonlinear Integer Programming. *Management Science,* 31(12), 1533-1546.

Hooker, J. N. (1988). Resolution vs Cutting Plane solution of Inference Problems: some computational experience. *Operations Research Letters,* 7,1(1988).

Jeroslow, R. G. and Lowe, J. K. (1984). Modelling with Integer Variables. *Mathematical Programming Study,* 22, 167-184.

Jeroslow, R. G. and Lowe, J. K. (1985). Experimental results on the New Techniques for Integer Programming Formulations, *Journal of the Operational Research Society,* 36(5), 393-403.

Jeroslow, R. E. and Wang, J. (1990). Solving propositional satisfiability problems, *Annals of Mathematics andAI,* 1,167-187.

Knopf F.C, Okos *MR,* and Reklaitis G.V. (1982). Optimal Design of Batch/Semicontinuous Processes, *Ind.Eng.Chem.ProcDesDev.,* Vol 21, No. 1, pp 79-86

Kocis, GJR. and Grossmann, I.E. (1987). Relaxation Strategy for the Structural Optimization of Process Flowsheets. *Industrial and Engineering Chemistry Research,* 26(9),1869-1880.

Kocis, G.R. and Grossmann, I.E. (1989). Computational Experience with DICOPT Solving MINLP Problems in Process Synthesis Engineering. *Computers and Chem. Eng.* 13, 307-315.

Kocis G.R., Giossmann LE. (1988) Global Optimization of Nonconvex MINLP Problems in Process Synthesis, *Ind£ngng.ChemJies.* 27, 1407-1421

Kondili E, Pantelides C.C and Sargent R.W.H. (1988). A General Algorithm for Scheduling Batch Operations, *Proceedings IntSymp. Process Syst.Eng.*, 3rd, pp62-75

Krarup, J. and Bilde, O. (1977). Plant Location, Set Covering and Economic Lot Size: An 0(mn) Algorithm for Structured Problems in L. Collatz et al. (eds), *Optimierung bei graphentheoretischen und ganzzahligen Problemen*, Int. Series of Numerical Mathematics, 36,155-180, Birkhauser Verlag, Basel.

Ku, H. and Karimi, I. (1988) Scheduling in Serial Multiproduct Batch Processes with Finite Intermediate Storage: A Mixed Integer Linear Program Formulation, *Ind. Eng. Chem. Res.* 27,1840-1848.

Ku, H. and Karimi, I. (1991) An evaluation of simulated annealing for batch process scheduling, *Ind. Eng. Chem. Res.* 30, 163-169.

Land, A. H., and Doig, A. G.(1960). An Automatic method for solving Discrete Programming Problems, *Econometrics* 28,497-520.

Lovacz, L. and Schrijver, A. (1989). Cones of Matrices and Set Functions and 0-1 Optimization, Report BS-R8925, Centrum voor Wiskunde en Informatica.

Magnanti, T. L. and Wong, R. T. (1981). Acclerated Benders Decomposition: Algorithm Enhancement and Model Selection Criteria, *Operations Research*, 29,464-484.

Martin, R.K. and Schrage, L. (1985). Subset Coefficient Reduction Cuts for 0-1 Mixed-Integer Programming, *Operations Research*, 33,505-526.

Mawekwang, H. and Murtagh, B.A. (1986). Solving Nonlinear Integer Programs with Large Scale Optimization Software. *Annals of Operations Research*, 5,427-437.

Miller D.L and Pekny J J[7]. (1991). Exact solution of large asymmetric traveling salesman problems, *Science*, 251, pp 754-761.

Nabar, S.V. and Schrage (1990). Modeling and Solving Nonlinear Integer Programming Problems. Paper No. 22a, Annual AIChE Meeting, Chicago, IL.

Nemhauser, G. L and Wolsey, L (1988). Integer and Cominatorial Optimization. Wiley, New York.

OSL Release 2 (1991) Guide and Reference, IBM, Kingston, NY.

Papageorgaki S. and Reklaitis G.V (1990a) Optimal Design of Multipurpose Batch plants-1. Problem Formulation , *Ind.Eng.ChemJZes.,Vol* 29, No. 10, pp 2054-2062

Papageorgaki S. and Reklaitis G.V (1990b). Optimal Design of Multipurpose Batch plants-2. A Decomposition Solution Strategy , *Ind.Eng.Chem.Res.,Vol* 29, No. 10, pp 2062-2073

Papageorgaki S. and Reklaitis G.V. (1990c). Mixed Integer Programming Approaches to Batch Chemical Process Design and Scheduling, ORSA/TIMS Meeting, Philadelphia.

Patel AJN., Mah R.S.H. and Karimi LA. (1991). Preliminary design of multiproduct noncontinuous plants using simulted annealing, *Comp & Chem Eng.* 15, 451-470
Pekny J.F and Miller D.L. (1991). Exact solution of the No-Wait Flowshop Scheduling Problem with a comparison to heuristic methods, *Comp & Chem. Eng.*, Vol 15, No 11, pp741-748.

Petersen C.C.(1991). A Note on Transforming the Product of Variables to Linear Form in Linear Programs, *Working Paper*, Purdue University.

Quesada I. and Grossmann I.E. (1992a). An LP/NLP based Branch anc Bound Algorithm for Convex MINLP Problems. To appear in *Comp. & Chem Eng.*

Quesada I. and Grossmann I.E. (1992b). Global Optimization Algorithm for Rational and Bilinear Progams. Manuscript in preparation

Rardin, R. L. **and** Choe, U.(1979). Tighter Relaxations of Fixed Charge Network How Problems, Georgia Institute of Technology, Industrial and Systems Engineering Report Series, #J-79-18, Atlanta.

Raman, R. **and** Grossmann, I. E. (1991). Relation between MILP modelling and Logical Inference for Process Synthesis, *Computers and Chemical Engineering,* 15(2), 73-84.

Raman, R. and Grossmann, I.E. (1992a). Integration of Logic and Heuristic Knowledge in MINLP Optimization for Process Synthesis, *Computers and Chemical Engineering,* 16(3), 155-171.

Raman, R. and Grossmann, I.E. (1992b). Symbolic Integration of Logic in Mixed-Integer Programming Techniques for Process Synthesis, submitted to *Computers and Chemical Engineering.*

Ravenmark D. and Rippin D.W.T. (1991). Structure and equipment for Multiproduct Batch Production, Paper No. 133a, Presented in AIChE annulal meeting, Los Angeles, CA

Reklaitis G.V (1990) Progress and Issues in Computer-Aided Batch Process Design, *FOCAPD Proceedings,* Elsevier, NY, pp 241-275

Reklaitis G.V. (1991). "Perspectives on Scheduling and Planning of Process Operations", *Proceedings Fourth Int.Symp. on Proc. Systems Eng.,* Montebello, Quebec, Canada.

Rich S*M* and Prokopakis GJ. (1986). Scheduling and Sequencing of Batch Operations in a Multipurpose **Plant,** *Ind.Eng.Chem.Res,* Vol. 25, No. 4, pp 979-988

Rich S.H and Prokopakis GJ. (1987). Multiple Routings and Reaction Paths in Project Scheduling, *Ind£ng.ChemJles,* Vol. 26, No. 9, pp 1940-1943

Sahinidis, N.V. and Grossmann, I.E. (1991a). MINLP Model for Cyclic Multiproduct Scheduling on Continuous Parallel Lines, *Computers and Chem. Eng.,* 15, 85-103.

Sahinidis, N.V. and Grossmann, I.E. (1991b). Reformulation of Multiperiod MILP Models for Planning and Scheduling of Chemical Processes, *Computers and Chem. Eng.,* 15,255-272.

Sahinidis, N.V. and Grossmann, I.E. (1991c). Convergence Properties of Generalized Benders Decomposition, *Computers and Chem. Eng.,* 15, 481-491.

Savelsbergh, M.W.P., Sigismandi, G.C. and Nemhauser, G.L. (1991) Functional Description of MINTO, a Mixed INTeger Optimizer, Georgia Tech., Atlanta.

Schrage, L. (1986). Linear, Integer and Quadratic Programming with LINDO, Scientific Press, Palo Alto.

SCICONIC/VM 2.11 (1991). Users Guide", Scicon Ltd, U.K.

Shah N. and Pantelides C.C., (1991). Optimal Long-Term Campaign Planning and Design of Batch Operations, *Ind. Eng. Chem. Res.,* Vol 30, No. 10, pp 2308-2321

Shah N., Pantelides C.C. and Sargent, R.W.H. (1991). Efficient Solution Techniques for Optimal Scheduling of Batch Operations. Working paper, Imperial College.

Sherali, H.D. and Alameddine, A. (1990) A new reformulation-linearization technique for bilinear programming problems, presented at ORSA/TIMS meeting Philadelphia

Sherali H. and Adams W.(1988) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, Technical Report, Virginia Polytechnic Institute./

Sherali, H. D. and Adams, W. P. (1989). Hierarchy of relaxations and convex hull characterizations for mixed integer 0-1 programming problems. Technical Report, Virginia Polytechnic Institute.

Sparrow RJ2, Forder GJ, Rippin D.W.T (197S) The Choice of Equipment Sizes for Multiproduct Batch Plant Heuristic vs. Branch and Bound , *Ind.Eng. ChemJProcDesJDev.*, Vol 14, No. 3, pp 197-203

Straub, D.A. and I.E. Grossmann (1992). Evaluation and Optimization of Stochastic Flexibility in Multiproduct Batch Plants, *Comp.Chem.Eng.*, 16, 69-87.

Suhami I. and Mah R.S.H, (1982) Optimal Design of Multipurpose Batch Plants, *Ind. Eng. Chem. Proc. Des. Dev.*, Vol 21, No. 1, pp 94-100

Sugden, SJ. (1992). A Class of Direct Search Methods for Nonlinear Integer Programming. Ph.D. thesis. Bon University, Queensland.

Swaney, R.E. (1990). Global solution of algebraic nonlinear programs. Paper No.22f, AIChE Meeting , Chicago,EL

Tomlin, J. A. (1971). An Improved Branch and Bound method for Integer Programming, *Operations Research*, 19,1070-1075.

Tomlin, J. A. (1988). Special Ordered Sets and an Application to Gas Supply Operations Planning. *Mathematical Programming*, 42,69-84.

Torres, F. E. (1991). Linearization of Mixed-Integer Products. *Mathematical Programming*, 49,427-428.

Van Roy, T. J., and Wolsey, L. A. (1987). Solving Mixed-Integer Programming Problems Using Automatic Reformulation, *Operations Research*, 35, pp.45-57.

Vaselenak J.A, Grossmann I.E. and Westerberg A.W. (1987). An Embedding Formulation for the Optimal Scheduling and Design of Multipurpose Batch Plants, *Ind.Eng.ChemJRes,26*, No1, ppl39-148

Vaselenak J.A, Grossmann I.E. and Westerberg A.W (1987) Optimal Retrofit Design of multipurpose Batch Plants, *Ind.Eng.Chem.Res*, 26, No. 4, pp718-726

Viswanathan, J. and Grossmann, I.E. (1990). A Combined Penalty Function and Outer-Approximation Method for MINLP Optimization. *Computers and Chem. Eng.* 14(7),769-782.

Voudouris V.T and Grossmann I.E. (1992a). Mixed Integer Linear Programming Reformulations for Batch Process Design with Discrete Equipment Sizes, *Ind. Eng. Chem. Res.*, 31, No.5, pp 1314-1326

Voudouris V.T and Grossmann I.E. (1992b). MILP Scheduling Model for Multipurpose Batch Plants. In preparation.

Wellons H.S and Reklaitis G.V. (1989). The Design of Multiproduct Batch Plants under Uncertainty with Staged Expansion , *Com . & Chem. Eng.*, 13, Nol/2, ppl 15-126

Wellons M.C and Reklaitis,G.V. (1991). Scheduling of Multipurpose Batch Chemical Plants. 1. Multiple Product Campaign Formation and Production Planning, *Ind.Eng.Chem.Res*, 30, No. 4, pp688-705

Williams, P. (1988). Model Building in Mathematical Programming. Wiley, Chichester.

Yuan, X., Piboleau, S., and Domenech, S. (1989). Une Methode d'Optimisation Non Linaire en Variables Mixtes pour La Conception de Precedes. RAIRO Recherche Operationnele