Mixed-Integer Nonlinear Programming Techniques For the Synthesis of Engineering Systems

by

Ignacio Grossmann

EDRC 06-83-90 ?

MIXED-INTEGER NONLINEAR PROGRAMMING TECHNIQUES FOR THE SYNTHESIS OF ENGINEERING SYSTEMS

Ignacio E. Grossmann

Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213

Keywords: engineering synthesis; systems integration; design optimization; mixed-integer nonlinear programming; mathematical modelling; design automation.

ABSTRACT

The development of new mixed-integer nonlinear programming (MINLP) algorithms, coupled with advances in computers and software, is opening promising possibilities to rigorously model, optimize and automate the synthesis of engineering systems. A general overview of the MINLP approach and algorithms will be presented in this paper with the aim of gaining a basic understanding of these techniques. Strengths and weaknesses will be discussed, as well as difficulties and challenges that still need to be overcome. In particular, it will be shown how proper problem representations, effective modelling schemes and solution strategies can play a crucial role in the successful application of these techniques. The application of MINLP algorithms in synthesis will be illustrated with several examples.

INTRODUCTION

Synthesis is one of the most important activities in the preliminary stages of engineering design. Synthesis deals with the question on how to select the topology and the parameters of a system that can meet specified goals, functional requirements and constraints. A major difficulty that arises in this problem is that commonly there is a very large number of design alternatives, and that selecting the appropriate topology and parameters for a design is a nontrivial task. The major reason for this is that at the synthesis stage decisions tend to have a very large impact in the cost, quality and downstream effects of a design (e.g. manufacturability, flexibility, constructability, etc.). Thus, there is clearly a major incentive to develop systematic synthesis methodologies that can effectively support design engineers for integrating improved systems.

In terms of systematic approaches to synthesis there are two major lines of attack that have emerged: (a) the heuristic approach which relies on intuition **and** engineering knowledge, (b) the optimization approach which relies on the use of mathematical programming techniques. The heuristic approach has the advantage of exploiting knowledge to simplify the problem and to quickly identify designs which are often of good quality. However, this approach does not offer any guarantee of optimality, mainly due to the fact that it cannot account for interactions and trade-offs. The optimization approach has the advantage of overcoming these limitations and of

providing a systematic framework that is domain independent with which one can model a large number of systems. The disadvantage with this approach, however, is that in principle it can lead to problems of large size that are difficult to solve. Furthermore, formulating problems within this framework is not always trivial. For a more extensive discussion on the merits of these approaches see for instance Stephanopoulos (1981) and Grossmann (1985).

Since the development of automated synthesis tools that can systematically examine large numbers of alternatives is currently being addressed with a knowledge based approach or with a mathematical programming approach, a relevant question to ask at this point is the following: Given advances that can be expected in the near future in Artificial Intelligence and Mathematical Programming, how will these impact the future development of the area of synthesis of engineering systems?

It is the objective of this paper to present a perspective on the above question from the Mathematical Programming viewpoint A major objective will be to show that recent algorithmic advances in mixed-integer nonlinear programming are starting to open promising possibilities to rigorously model, optimize and automate synthesis problems. Examples will be drawn from the chemical engineering domain where these techniques have recently found many successful applications.

This paper will be organized as follows. We will first outline the major steps that are involved in the mathematical programming approach: formulation of a superstructure of alternatives, and modelling and solution of a mixed-integer nonlinear programming (MINLP) problem. We will discuss for the former step major alternative representations that can be used, and identify the limitations that are present at this point. We will then concentrate on the modelling and on the basic ideas behind several MINLP algorithms that have emerged. The importance of solution strategies will be discussed and illustrated with a number of example problems. Finally, throghout the paper we will briefly address the question of combining the quantitative and qualitative approaches to synthesis.

MATHEMATICAL PROGRAMMING APPROACH

From a conceptual viewpoint, the synthesis problem can be stated as follows. Given art goals, functional specifications and constraints of a design that is to be synthesized from a set of known or available components. The problem then consists in integrating a system that meets the functional specifications and constraints while optimizing a given objective or goal.

Examples of synthesis problems in different domains include the design of structures in civil engineering, of process flowsheets in chemical engineering, of VLSI circuits in electrical engineering, and of servomechanisms in mechanical engineering. Note that these problems are mainly concerned with the question on how to integrate a system given known or available components.

A major feature in synthesis problems of engineering systems is that they involve the selection of a configuration or topology, as well as its design parameters. That is, one has to determine on **the** one hand which components should integrate a system and how they should be interconnected; on the other hand one has to determine the sizes and parameters of the components. The former clearly imply making discrete decisions, while the latter imply making a choice from among a continuous space. Thus, from a conceptual standpoint the synthesis problem corresponds to a discrete/continuous optimization problem which mathematically gives rise to an MINLP problem.

In general, the major steps involved in the MINLP approach are as follows:

Step 1. A superstructure is postulated that has embedded alternatives that are candidates for a feasible and optimal design.

Step 2. The superstructure is modelled as the MINLP problem:

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

Here y represents a vector of 0-1 variables that denote the potential existence or selection of components (0 not selected, 1 selected), while x represents a vector of continuous variables which correspond to sizes and parameters. C(y,x) represents the objective function (e.g. weight, cost), h(y,x) = 0 the performance or analysis equations and g(y,x) £ 0 the design specifications and logical constraints. For many engineering applications in synthesis the dominant structure is that the MINLP is most often linear in the 0-1 variables with nonlinearities being present in the continuous variables.

Step 3. The optimal design is obtained from the superstructure by solving the corresponding MINLP problem. The optimal topology is defined by those components which were not "deleted" from the superstructure (i.e. with values of 1 for the binary variables).

It should be noted that for the most general case in the above approach nonlinearities are involved in the optimization model of step 2 which gives rise to an MINLP problem. However, when simplified synthesis models are developed it is possible that the MINLP might reduce to a simpler form. For instance, if all the equations involved are linear this gives rise to a mixed-integer linear program (MILP). If no 0-1 variables are used for modelling the selection of components, but only continuous variables are used, this will give rise to either a nonlinear program (NLP) or to a linear program (LP) depending whether nonlinearities are present or not. When the model reduces to any of these simpler forms, or even more, to a specialized class of

problems (e.g. network flow problems), a number of standard optimization techniques arc available which can properly exploit the structure of the problem. These for instance include the simplex algorithm and interior point methods for LP (Goldfarb and Todd, 1989), the branch and bound method for MHJP (Wolsey and Nemhauser, 1988), specialized combinatorial optimization techniques (Wolsey and Nemhauser, 1988), and the reduced gradient method (Murtagh and Saunders, 1985) or the SQP algorithm for NLP (Han, 1977; Powell, 1977).

It should also be noted that for the case when sizes of the components are available in discrete sizes these can be expressed in terms of 0-1 variables. Therefore for convenience in the presentation we will treat this as a particular case and assume that sizes are available in continuous form.

While in the past synthesis problems often had to be simplified to avoid the explicit solution of an MINLP, it is not until very recently that these simplifications have no longer been required due to new algorithmic developments. Among the first applications that we can cite where synthesis problems were modelled as an MINLP are the synthesis of gas pipelines (Duran and Grossmann, 1986a) and the synthesis of process flowsheets by Kocis and Grossmann (1987).

The two crucial steps in the approach described above arc Step 1 for generating the superstructure, and Step 3 for solving the MINLP problem. As it turns out, however, Step 2 is also extremely important because the way one models MINLP problems can have a great impact on the performance of the algorithms. The next section will discuss the question of formulation of superstructures.

SUPERSTRUCTURES

As indicated in the previous section, in order to formulate the synthesis problem as an optimization problem one has to develop a representation containing all the alternative designs that are to be considered as candidates for the optimal solution. Developing an appropriate superstructure is clearly of paramount importance, as the optimal solution that one obtains can only be as good as the representation that is being used. While this point is often regarded as a major weakness of the optimization approach, knowledge based approaches suffer also from a similar limitation since these also require a representation of alternatives which is often implicit in nature.

Superstructure representations can in general be explicit or implicit in nature. The former give in general rise to networks, while the latter give rise to trees. As an example, consider the separation of a single feed consisting of 4 chemicals A,B,C, and D that is to be separated into pure products. Here A is the most volatile product while D is the least volatile product. Since distillation is the technology considered with "sharp" splits, only adjacent products in volatility can be separated. The 5 different alternative sequences consisting of all different processing components can be represented through the tree shown in Fig. 1 (Hcndry

and Hughes, 1972). Each node in this tree represents a processing component which physically corresponds to a separator (distillation column). For instance, the separator A/BCD is a distillation column that separates chemical A from the mixture BCD. Since A is more volatile it is removed at the top of the distillation column while the mixture BCD is removed at the bottom. Also note from Fig. 1 that one feasible sequence is given by A/BCD, B/CD, C/D; i.e. the most volatile chemical is removed at each step.

The tree representation in Fig. 1, which can be readily generalized for mixtures with a larger number of chemicals, lends itself to decomposition. Here the discrete alternatives of selection of separators can be enumerated implicitly through a branch and bound search in order to find the sequence with minimum total cost. However, in using this tree representation the MINLP problem must be converted into the separable optimization problem over the discrete space $Y_{D_i}, i=1..m$,

$$\begin{aligned} Z &= \min_{y_i} \sum_{i=1}^{m} C_i(y_i) \\ \text{s.t.} &\sum_{i=1}^{m} g_{ij}(y_i) \leq a_j \quad j=1,...t \\ & y_i \in Y_{D_i}, \quad i=1...m \end{aligned} \tag{1}$$

where continuous variables are selected independently at each node of the tree. For the case when there are significant interactions among the nodes (e.g. when heat integration is considered among the columns), the formulation in (1) is likely to lead to suboptimal solutions even if the branch and bound search is performed rigorously. Note also that in the AI approach, which relies on an implicit enumeration of the discrete alternatives, fewer nodes are examined in the tree with the use of heuristics (Lien et al., 1987; Maher, 1988). This, however, increases further the likelihood of obtaining suboptimal solutions.

On the other hand, consider the network representation shown in Fig. 2 where the alternatives of Fig.1 have been superimposed (Andrecovich and Westerberg, 1985). Here in contrast to the tree, every node corresponds to a distinct separator. Thus, 10 nodes for the processing components are required instead of the 13 in Fig. 1 where the separators A/B, B/C and C/D are represented twice. Note that in the network of Fig. 2 alternative sequences can be represented by using the same subset of nodes (e.g. Sequence 1: A/BCD, B/CD, C/D, Sequence 2: AB/CD, A/B, C/D share the node C/D). This network is a more compact representation for modelling the problem explicitly as an MINLP. The advantage of this model is that interactions can be explicitly accounted for in the optimization since the continuous variables can be optimized simultaneously with the selection of the configuration which is obtained by "deleting" nodes and streams that are not required for the optimal structure. The disadvantage with the MINLP model, however, is that one loses the capability of performing the straightforward decomposition that is possible with the tree. It will be shown later in the paper, however, that one

can still resort to more sophisticated decomposition schemes to rigorously solve the MINLP problem for the network.

From the above discussion, it follows that network representations that arc explicitly modeled as MINLP problems provide a more general and rigorous framework for the optimization. As an additional example of a network superstructure consider the synthesis of the structure shown in Fig 3.a that involves three loads. A possible network representation of the alternative topologies is shown in Fig. 3.b. In this network the potential existence of symmetric bars has been embedded which gives rise to many alternative topologies which are obtained by deleteing different combinations of bars.

The next question to be addressed is how to actually postulate or derive the superstructures. This is an easier task for homogeneous systems that consist of similar components (e.g. structures, heat exchanger networks, distillation sequences) than for heterogeneous systems that consist of a variety of different component types (e.g. VLSI circuits, process flowsheets, electromechanical devices). Examples of homogeneous superstructures arc the networks shown in Fig. 2 and Fig. 3.b.

As an additional example of an homogeneous system, consider the synthesis of heat exchanger networks which is one of the synthesis problems that has been studied most extensively in the chemical engineering literature (see Gundersen and Naess, 1987, for a review). Virtually every known approach has been tried on this synthesis problem. The objective in this problem is to synthesize a network of heat exchanger networks where heat from hot streams can be transferred to cold streams at minimum cost. The major cost elements are the cost of the heat exchangers and the cost of heating and cooling utilities. One possible representation for this problem that allows for stream splitting and mixing is shown in Fig. 4, where each exchanger unit corresponds to a potential match of pairs of streams (sec Yee and Grossmann, 1989). A more general and richer representation for the same problem where the layout and pipe connections can be accounted for, is shown in Fig. 5 where one only has to specify the maximum number of heat exchanger units in the network (see Yee and Grossmann, 1988). Note that in this case aside from all the alternatives of Fig. 4, there is even the possibility of mixing different process streams or to represent multi-stream heat exchangers.

It is interesting to note in the previous example, that in the superstructure of Fig. 4 there is a one-to-one correspondence between the components and the function they can perform (i.e. each exchanger involves a specific match of a hot stream H and a cold stream C). On the other hand in the superstructure of Fig. 5 there is a one-to-many relationship between the components and their functions (e.g. exchanger 1 can perform matches Hl-Cl or H1-C2). Another example of a one-to-many relationship, is the superstructure for separation in Fig. 6 proposed by Sargent and Gaminibandara (1976). Note for instance that column 1 can perform any of the three "sharp" separations of the feed (A/BCD, AB/CD, ABC/D), or if needed a "non-sharp" separation wherc chemicals distribute at the top and bottom of the cut The network in Fig. 6, in addition to accomodating sharp and non-sharp splits, has embedded thermally integrated columns (e.g.

Petlyuk columns) as alternative designs. From these examples it is clear that superstructures that have one-to-many relationships between components and functions are richer in terms of alternatives that they have embedded. Furthermore, in these type of superstructures geometrical aspects of the problem are more readily accounted for (e.g. layout). On the other hand, the more restricted one-to-one superstructures (e.g. Figs. 2, 3.a and 4) tend to require simpler MINLP models that are quicker to solve.

To systematically develop superstructures for heterogeneous systems is in principle a more difficult task. For instance consider a chemical process flowsheet that is composed of reaction, separation and heat integration subsystems. One could in principle develop a superstructure by combining the superstructures for each subsystem. This, however, could lead to a very large MINLP optimization problem.

One option to reduce the size of the optimization problem is to use higher level representations of superstructures which aggregate the components and streams of the detailed superstructures. Here the example par excellence is the simultaneous synthesis of a flowsheet and the heat exchanger network. The detailed superstructure of the latter can be aggregated through targets for minimum utility cost and minimum number of units which can be modelled respectively as LP and MILP transportation (Cerda and Westerberg, 1983) or transshipment problems (Papoulias and Grossmann, 1983). For the case when the stream data in the flowsheet are variable (flowrates and temperatures), the utility target problem can be modelled as a system of nonlinear inequalities (see Duran and Grossmann, 1986c), while for the case when only the flowrates are variables it can be represented by the transportation or transshipment equations (see Papoulias and Grossmann, 1983; Andrecovich and Westerberg, 1985). These representations clearly simplify the synthesis problem. They allow a great reduction in the size of the optimization problem since their corresponding superstructures are replaced by relatively few algebraic equations and inequalities that are added to the MINLP of the process flowsheet. The drawback, however, is that these aggregated models do not provide all the explicit information of the heat exchanger network which must then be synthesized at a second stage.

Finally, another example of aggregation to account for downstream concerns are the scheduling models that have been developed by Birewar and Grossmann (1989) for the design of multiproduct batch processes. Here, these authors have been able to develop linear constraints for flowshop scheduling that can be readily be incorporated within mathematical formulations to optimize the sizing and structure of these plants. These linear models introduce only a small error in the estimation of time requirements while accounting for all the possible sequences for scheduling.

Another approach to reduce the size of the MINLP is to assume that some preliminary screening is performed (e.g. through heuristics) in order to postulate a smaller number of alternatives in the superstructure (Kocis and Grossmann, 1987). While this approach would seem to be restrictive, it does provide a systematic framework for analyzing specific alternatives at the level of tasks or functions. As an example, consider the synthesis of a chemical process where a

preliminary screening would indicate that the major options are as follows: single or two stage compression for the feed, three possible reactor types, possible use of membrane separator for the purge stream, use of flash separation with the option of absorption/distillation columns. Fig. 7 displays the superstructure for these alternatives. This superstructure has actually embedded a minimum of 24 different configurations. As another example, Fig. 14a shows a superstructure for the HDA process for manufacturing benzene. This superstructure was developed by Kocis and Grossmann (1988b) based on the alternatives that were postulated by Douglas (1988) in the hierarchical decomposition procedure.

Thus, generating superstructures for heterogeneous systems based on specific alternatives at the level of functions is actually not a very difficult problem provided there is sufficient qualitative information on the main alternatives. However, it is clear that in order to consider a larger number of alternatives, the more natural approach would be to combine all the superstructures of the homogeneous subsystems. At this point, it is an open question as to whether this will require the capability of solving much larger MINLP problems, the development of strategies to successively aggregate and disaggregate subsystems, or else a combination with AI techniques (e.g. hierarchical decomposition) to systematically eliminate alternatives from a superstructure that potentially has a very large number of alternatives.

MINLP MODELLING OF SUPERSTRUCTURES

Having developed a superstructure for the candidate designs to be considered, the next step involves the modelling of the MINLP optimization problem. The major feature in such models is the modelling of discrete decisions which is typically performed with 0-1 variables. For most applications it suffices to assign these variables to each potential component in the superstructure as the interconnections are activated or deactivated according to the selection of units. There are, however, cases when it is also necessary to assign 0-1 variables to the interconnections.

The handling of 0-1 variables allows the specification of constraints which are extremely relevant for synthesizing practical engineering systems. These constraints, which cannot be handled with continuous optimization techniques, offer the possibility to the designer to impose restraints and control the complexity of the toplogy that is to be synthesized. Typical examples include the following:

a) Multiple choice constraints for selecting among a subset of components I:

Select only one component: $\sum_{j} y_j = 1$ (2)

Select at most one component: $\sum_{j} y_j \leq 1$ (3)

Select at least one component: $\sum_{j} y_j \geq 1$ (4)

b) If then conditions:

If component k is selected then component i must be selected:

$$y_k - y_i \leq 0 \quad (5)$$

Note that if y_k=l (component k selected), the inequality forces yj=l (component i is selected); however, the converse is not true. In addition, 0-1 variables can be used to activate or deactivate continuous variables, inequalities or equations. As an example consider the following logical condition for the continuous variable x, where L and U are lower and upper bounds, respectively:

$$\text{If} \quad y = 1 \rightarrow L \leq x \leq U, \quad \text{if } y = 0 \rightarrow x = 0$$

This condition can be modelled through the constraint

$$Ly \leq x \leq Uy \quad (6)$$

The above constraint is often used in conjunction with cost models with fixed cost charges which again requires the use of 0-1 variables (see Garfinkel and Nemhauser 1972, Nemhauser and Wolsey, 1988). Furthermore, it has been recently shown by a number of authors (e.g. Cavalier and Soystcr, 1987) that virtually any prepositional logic statement can be systematically translated into a set of linear inequalities involving 0-1 variables. A recent paper by Raman and Grossmann (1990), shows how these ideas can be applied in the domain of chemical engineering to model inference problems, design production systems as well as complex discrete constraints that arise in design problems.

While the use of 0-1 variables introduces a very important capability in the modelling of MINLP problems, it is also true that the way one models an MINLP can have a great impact on the performance of the MINLP algorithm. This phenomenon has been widely recognized in the areas of integer programming (e.g. see Williams, 1978; Nemhauser and Wolsey, 1988) and nonlinear programming (Papalambros and Wilde, 1988). As an example consider the logical constraint that may arise in a multiperiod design/manufacturing problem:

$$\sum_{i}^{m} y_i - mz \leq 0 \qquad (7)$$

where z represents the selection of a given component and y_i the operation of the component in periods i, i=l,...m. This constraint simply states that if z=0 no operation of the component is possible in the m time periods, while if z=l operation in any of the m periods is possible. While (7) is a "legitimate" constraint, it turns out that its equivalent representation by the set of inequalities

$$y_i - z \leq 0 \quad i=l,...m \qquad (8)$$

is a much more effective way to model this constraint since its relaxation with continuous variables y_4 corresponds to the convex hull of 0-1 solutions (sec Fig. 8). Despite the fact that one requires a larger number of constraints in (8), this introduces a greater number of extreme points with 0-1 values in the linear programming relaxation which greatly reduces the enumeration in a branch and bound procedure. Another typical example in modelling is the use of a tight upper bound U in the logical constraint, x - Uy ≤ 0, to tighten the relaxation problem where the 0-1 variables are treated as continuous.

Some of these empirical observations have led to the theoretical study in integer linear programming of facets of 0-1 polytopes that define the convex hull of integer programming problems (Schrijver, 1986). Algorithms are starting to emerge which can systematically generate approximations to these type of constraints, and hence reformulate a "badly" posed integer programming problem in order to tighten the continuous relaxation (e.g. see Crowder *et aU* 1983, for unstructured 0-1 linear problems, Van Roy and Wolsey, 1987, for MILP problems). The objective in these reformulations is to ideally be able to solve the integer linear programming problems as LP problems where the 0-1 variables are relaxed as continuous variables in order to avoid the combinatorial search of a branch and bound procedure.

In order to appreciate the great impact of modelling in integer programming consider the network superstructure in Fig. 9 that involves the selection of 38 processes and 25 chemicals that are to be integrated for an industrial chemical complex (Sahinidis and Grossmann, 1989a). Since variable prices and demands were considered over a 10 year horizon, 4 time periods were considered to model these changes. A straightforward optimization model led to an MILP involving 152 0-1 variables, 709 continuous variables and 785 constraints. After 7 hours of CPU-time on an IBM-3090 computer the branch and bound method had not terminated the seach and was only able to find a suboptimal solution within 6% of the optimum. A major reason for this had to do with the fact that the linear programming relaxation is not very tight However, as shown by Sahinidis and Grossmann (1989a), the MILP model can be reformulated by using a variable disaggregation scheme based on the lot sizing problem to strengthen the relaxation. For this example the reformulation was larger in terms of constraints as it involved 152 0-1 variables, 1037 continuous variables and 1431 constraints (i.e. almost twice the number of constraints).

The important point, though, was the fact that the CPU-time required to solve the problem to optimality was 17 minutes, leading to an order magnitude reduction in the computer time!

As opposed to the integer linear programming case, in MINLP there is the additional complication that nonlinearities can also be often formulated in many different ways which are equivalent, and as expected this can also have a great impact on the performance of MINLP algorithms. In general, three major empirical guidelines for a "good" MINLP formulation are the following:

1. Try to keep the problem as linear as possible.
2. Try to develop a formulation that has as tight an NLP relaxation as possible.
3. If possible, reformulate the MINLP as a convex programming problem.

The motivation behind these guidelines requires some basic understanding of the MINLP algorithms which we will cover in the next section.

MINLP ALGORITHMS

BASIC ALGORITHMS

While there is a vast body of literature on LP, NLP, and on integer LP with special structures, this is not the case for MINLP. The reason is that MINLP is regarded as a very difficult problem since it corresponds to an NP-complete problem (Nemhauser and Wolsey, 1988) that is prone to combinatorial explosion for large problems. In our view, however, it is a mistake to regard these problems as "unsolvable". Not only are the applications for MINLP extremely rich, but with current methods and technology one can in fact already solve problems of significant size and complexity as will be shown later in the paper. Futhermore, with advances in new algorithms and computer architectures it is reasonable to assume that over the next decade we will see increases in the order of magnitude of sizes of problems that can be currently solved. A recent example of this trend is the parallel algorithm for the asymmetric traveling salesman problem by Pekny and Miller (1989) who have been able to solve up to 7,000 city problems to optimality in less than 30 minutes of computer time. These problems, which correspond to linear integer programming problems that are highly structured, involve up to 14,000 constraints and 50,000,000 variables.

Our objective in this section will be to provide a general overview of the basic MINLP algorithms, and emphasize their basic ideas and properties. Firstly, for convenience in the presentation we will assume that the MINLP has the restricted form where the 0-1 variables appear in linear form and where no equations are involved. Nonlinearities for the continuous variables are assumed to be present only in the objective function and in the inequality constraints. The formulation of the MINLP is then as follows:

$$Z = \min\ c^T y + f(x)$$
$$\text{s.t}\ By + g(x) \leq 0 \qquad \text{(PI)}$$
$$y \in Y\ ,\ x \in X$$

where $Y=\{y \mid Ay \leq a\ , y \in \{0,1\}^m\}$, $X=\{x \mid x^L \leq x \leq x^u\ ,\ x \in R^n\}$

Major algorithms for solving the MINLP problem in (PI) include the following:

a) Branch and bound (Beale, 1977; Gupta, 1980; Gupta and Ravindran, 1983)

b) Generalized Benders Decomposition (Benders, 1962; Geoffrion, 1972)

c) Outer-Approximation (Duran and Grossmann, 1986b)

The branch and bound method for MINLP is a direct extension of the linear case (MILP). This method starts by relaxing the integrality requirements of the 0-1 variables which leads to a continuous NLP optimization problem. It then continues by performing a tree enumeration where a subset of 0-1 variables are successively fixed at each node. The solution of the corresponding NLP at each node provides a lower bound for the optimal MINLP objective function value. This lower bound can then be used to expand the nodes in a breadth first enumeration (i.e. expand the node with lowest lower bound), or else to fathom nodes in a depth first enumeration whenever the lower bound exceeds the best current upper bound. Clearly the size of the tree is dependent of the number of 0-1 variables (maximum of 2^{m+1}-1 nodes), although of course the objective in the search is to hopefully enumerate only a small subset of nodes. Fig. 10 presents an example of a branch and bound search for a minimization problem involving three 0-1 variables. Note that at the root node of the tree, where the 0-1 variables are treated as continuous variables that lie between 0 and 1, Z=5.8 yielding the noninteger point [0.2,1,0]. To find the optimal integer solution a breadth first enumeration is used to locate the optimum in node 9 with an optimal objective value of Z=8. The optimal value of the 0-1 variables is [0,1,1]. Note that in this case out of the IS nodes that are possible in the tree, 9 nodes had to be examined to locate the optimum solution.

The major disadvantage of the branch and bound method is that it may require the solution of a relatively large number of NLP subproblems which cannot be updated as readily as in the linear case where few pivot operations are required to update the LP solution of a new node. On the other hand, if the MINLP has a tight NLP relaxation the number of nodes to be enumerated may be modest. In the limiting case where the NLP relaxation exhibits 0-1 solutions for the binary variables (convex hull formulation) only one single NLP problem at the root node need to be solved.

In contrast to **the** branch and bound method, both the Generalized Benders Decomposition (GBD) and Outer-Approximation (OA) algorithm consist of solving at each major iteration an NLP subproblem (with all 0-1 variables fixed) and an NfILP master problem as shown in Fig. 11. The NLP subproblems have the role of optimizing the continuous variables and provide an upper bound to the optimal MINLP solution. The MILP master problems have the role of predicting a lower bound to the MINLP as well as new 0-1 variable values for each major iteration. The predicted lower bounds increase monotonically as the cycle of major iterations proceeds, and the search is terminated when the predicted lower bound coincides or exceeds the current upper bound.

It should be noted that the logic used in the search of both the GBD and the OA method provides an appealing interpretation for synthesis problems. Both involve an iterative search over different topologies in which the master problem makes the discrete choices for the topologies based on the information that is accumulated from the parametric optimization of the configurations that are examined at each iteration. Furthermore, the master problem has the capability of terminating the search based on the lower bound that it predicts.

The main difference between the GBD and OA method lies in the definition of the MILP master problem. In the case of GBD it is given by a dual representation of the continuous space, while in the case of the OA it is given by a primal approximation. In particular, given solutions x^k with multipliers n^k of the NLP subproblems for fixed y^k, k=l,...K,

$$\begin{aligned} Z(y^k) = \min\ & c^T y^k + f(x) \\ \text{s.t. } & g(x) \pounds - B y^k \\ & x \in X \end{aligned} \tag{9}$$

the master problem of GBD is given by

$$z\pounds_B = \min_{y^0 \text{ oa}} a_{GB} \tag{10}$$

$$\text{s.t.} \quad a_{GB} * c^T y + f(x^k) + \pounds\ jij\ [_{gj}(x^k) - b - y] \qquad k=l..K$$

$$y_\epsilon Y \ , \ a_{GB}\ eR^1$$

while the master problem of OA is given by

$$Z_{OA}^{K} = \min_{y,x,\alpha_{OA}} \alpha_{OA} \qquad (11)$$

$$\left.\begin{aligned} & s.t.\ c^T y + f(x^k) + \nabla f(x^k)^T (x - x^k) - \alpha_{OA} \leq 0 \\ & \\ & By + g(x^k) + \nabla g(x^k)^T (x - x^k) \leq 0 \end{aligned}\right\} \quad k=1..K$$

$$y \in Y,\ x \in X,\ \alpha_{OA} \in R^1$$

Note that in both methods the MILP master problems in (10) and (11) accumulate new constraints as iterations proceed. GBD accumulates one Lagrangian cut in the space of the 0-1 variables, while OA accumulates a set of linear approximations of the nonlinear constraints in the space of both the 0-1 and continuous variables. It can be actually proved (see Duran and Grossmann, 1986b), that each Lagrangian cut in GBD represents a surrogate constraint of the corresponding linear approximations in OA. Therefore, since the master problem of the OA method is richer in information, it can be proved that it predicts stronger lower bounds than GBD and therefore it requires fewer major iterations (Duran and Grossmann, 1986b). This then implies that the OA method requires the solution of fewer NLP subproblems (i.e. fewer parametric optimizations for given topologies), which in addition become successively easier to solve as the MILP master problem also predicts values of continuous variables which provide excellent initial guesses (Kocis and Grossmann, 1989a). This property is especially significant in synthesis models where the nonlinear optimization is expensive to evaluate (e.g. through a process simulator or through a finite element analysis).

On the other hand, it is also clear that the computational demands on the MILP master problem of OA are greater since when compared to the master of GBD, it contains the continuous variables as well as a larger number of constraints. The advantage in GBD is that its master problem contains only the 0-1 variables and one scalar variable as well as fewer constraints.

In terms of convergence, neither GBD nor OA have the property that the convex hull formulation converges in **one** single major iteration as would be the case in the branch and bound method Here instead the OA algorithm converges in one major iteration if the MINLP reduces to an MILP as then the master problem in (11) provides an exact representation of the original problem. In the case of GBD, convergence cannot be guaranteed in one major iteration for this limiting case. However, as has been shown by Magnanti and Wong (1981), the optimal formulation for GBD in terms of number of major iterations is the convex hull formulation for which in practice convergence is often achieved in few iterations. Further, a distinct advantage with GBD is that special structures can be exploited more readily. For instance, fixing a subset of complicating variables the resulting subproblem might reduce to an LP, to a convex NLP or to a set of disjoint problems that can be solved in parallel (Geoffrion, 1972).

As for sufficient conditions for convergence to the global optimum, all the above algorithms require that the functions in (MINLP) satisfy some form of convexity conditions. The specific requirements vary with each algorithm. For instance, since the OA algorithm is based on the construction of supporting hyperplanes with function linearizations, strict convexity is required by the functions that involve the continuous variables. On the other hand, the branch and bound method requires that each of the NLP subproblems have a unique solution, and therefore strict convexity is not required. Finally, in the case of GBD strict convexity is required for fixed values of the binary variables, and quasi-convexity for the Lagrangian function in terms of the 0-1 variables (Geoffrion, 1972). Note, that this condition is not as stringent as for the OA algorithm.

To provide some insight into the computational performance of GBD and the OA method, Table 1 presents results for four convex synthesis problems that involve up to 40 0-1 variables, 38 continuos variables and 142 constraints. As can be seen, the OA method always requires fewer iterations, and the differences in CPU-time are most significant in the largest problem. Note however, that the GBD method does not necessarily require longer times since its MILP master problem is faster to solve. This is the case of the third problem in Table 1.

EXTENSIONS

The three MINLP algorithms described above can be extended to explicitly handle nonlinear equations $h(x) = 0$. In the case of branch and bound this is simply accomplished by appending these equations to the relaxed NLP subproblems that are solved at each node. For the case of GBD no modification is required in the master problem (10), as the multipliers μ of the NLP subproblem with the equations will reflect their effect. In the case of the OA algorithm, handling of equations in the master problem can be accomplished with the equality relaxation strategy by Kocis and Grossmann (1987) (OA/ER algorithm). Here, linearizations of the equations at the solution of the NLP subproblem k, are added to the master problem in (11) by relaxing them according to the sign of the Lagrange multipliers; that is,

$$T^k \left[\nabla h(x^k)^T (x - x^k) \right] \leq 0 \qquad (12)$$

where $T^k = [t_{ii}^k]$ and

$$t_{ii}^k = \begin{cases} 1 & \text{if } \lambda_i^k > 0 \\ -1 & \text{if } \lambda_i^k < 0 \\ 0 & \text{if } \lambda_i^k = 0 \end{cases}$$

As has been shown by Kocis and Grossmann (1987) sufficient conditions for global optimality with the OA/ER algorithm require quasi-convexity of the relaxed nonlinear equations.

In addition to the above cited algorithms, a number of extensions have been suggested recently. Among these the treatment of nonconvexities is perhaps the most important since engineering models are often not convex.

In the case of GBD, Floudas *et al* (1988) have proposed strategies to partition continuous variables that are involved in bilinear terms (which are nonconvex) into complicating variables that are introduced in the master problem, and noncomplicating variables that are optimized in the NLP subproblem. In this way, MINLP problems that are linear in the 0-1 variables and bilinear in the continuous variables can be decomposed into continuous LP subproblems (fixed 0-1 and fixed complicating continuous variables) and MILP master problems (involving the 0-1 and complicating continuous variables). Global optima can then be obtained for each, the LP subproblem and the MILP master, respectively. However, theoretically this does not imply that the global optimum of the MINLP can be attained since the lower bound from the master might not always be valid due to nonconvexities that are introduced in the Lagrangian function. Nevertheless, computational results reported by Ciric and Floudas (1988a) and Floudas *et al* (1988) seem to indicate that the success ratio is high which is most probably due to the loose approximations in the master problem with which a larger number of integer points is examined.

As for the case of the OA/ER algorithm, Kocis and Grossmann (1988) proposed a two phase strategy in which nonconvexities are identified numerically with local and global tests in the first phase where the OA/ER algorithm is applied. In the second phase, linearizations of the constraints that are identified as being nonconvex, are relaxed with slack variables which are introduced with a penalty function in the master problem. Also, at this stage, since the master problem is not guaranteed to predict rigorous lower bounds, the termination criterion is changed to one where the cycle of major iterations continues until the NLP subproblem fails to decrease the objective function value. This strategy was shown to yield the global optimum in 80% of a set of test problems.

Recently, Viswanathan and Grossmann (1989) have developed a new variant of the OA/ER algorithm that makes use of an augmented penalty function in the master problem (AP/OA/ER algorithm). The algorithm does not require that an initial guess of the 0-1 variables be supplied as it starts by solving the relaxed NLP problem. If an integer solution is found the algorithm stops. Otherwise it proceeds to formulate an MILP master problem that is similar in nature to the phase two master by Kocis and Grossmann (1988). However, instead of trying to identify nonconvex linearizations, slacks are added to all the linearizations of the nonlinear relaxed equations and inequalities, yielding the master problem:

$$
\begin{aligned}
Z^{K}_{AP} = \min_{y,x} \; & \alpha_{OA} + \sum_{k=1}^{K} \left[w_q^k q^k + (w J) V + (w J)^T s^k \right] \\
\text{s.t.} \; & c^T y + f(x^k) + V/(x^k)^T (x - x^k) - a_{0A} \le q^k \\
& 1^* f V h(x^k)^T (x - x^k)] \le p^k \\
& By + g(x^k) + V g(x^k)^T (x - x^k) \le 0 \\
& y \in Y \;, \; x \in X, \; \alpha_{OA} \in R^1 \\
& q^k \;, \; p^k \;, \; s^k \ge 0 \;, \qquad k = 1..K
\end{aligned}
\qquad \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \quad k = 1,..K \qquad (13)
$$

where q^k, p^k, s^k are slack variables with corresponding large finite weights w_q^*, w£, w_i^*.

For the general case, the cycle of major iterations in AP/OA/ER proceeds until there is no decrease in the NLP solution. It is interesting to note, however, that if the convexity conditions are satisfied, the MILP master problem predicts rigorous lower bounds and in this case this algorithm reduces to the OA/ER algorithm, except that it uses as a starting point the solution of the relaxed NLP. Computational experience with this method has shown a high degree of robustness with nonconvex problems.

Among other important extensions, Yuan *et al* (1987) have extended the OA algorithm for the case when the 0-1 variables are nonlinear. This is simply accomplished by linearizing the 0-1 variables in the master problem. Convergence to the global optimum can be guaranteed for the case when these functions are strictly convex. Loh and Papalambros (1989) have developed a variant of the outer-approximation algorithm in which only the most recent linearization is kept for the master problem. Although theoretically this scheme does not guarantee convergence, it enhances the chances of overcoming nonconvexities. Furthermore, these authors applied this technique to MINLP problems that involve sets of discrete variables rather than 0-1 variables, and showed that good performance can be obtained.

For the branch and bound method, Ostrovsky *et al* (1989) have proposed a strategy in which, at each node corresponding to the best bound, the relaxed NLP solution is rounded in order to compute an upper bound. The search is terminated when the lower and upper bounds lie within a specified tolerance. This strategy, however, would seem to be only suitable for the case when there is a small gap in the relaxed NLP solution. Finally, Mawengkang and Murtagh (1986) and Mawengkang (1988) have proposed a feasibility technique where the main idea is to round the relaxed NLP solution to an integer solution with the least local degradation. This is accomplished through the computer code MINOS by successively forcing superbasic variables to become nonbasic based on information of the reduced costs. While this method has no guarantee of global optimality, it has shown to have very good performance on a set of test problems with modest computational effort (typically 50% of time over the relaxed NLP problem).

COMPUTATIONAL EXPERIENCE

While until very recently there was very little experience reported in the literature for solving MINLP problems, this situation has undergone a significant change over the last few years with developments in algorithms for MINLP and computer software for NLP (Murtagh and Saunders, 1985; Han, 1976; Powell, 1976; Vasantharajan *et aU* 1989), MILP (MPSX, SCICONIC, ZOOM), and modelling systems such as GAMS (Brooke *et* a/, 1988) which have facilitated the implementation of MINLP algorithms (e.g. Paules and Floudas, 1989; Kocis and Grossmann, 1989a).

For instance, Table 2 presents computational results with DICOPT++ (Viswanathan and Grossmann, 1989) where the AP/OA/ER method has been implemented as part of the modelling system GAMS. In DICOPT++ the MINLP is specified in equation form and the user need not be concerned with the deatails of the algorithm which uses MINOS for the NLP optimization and MPSX for the MILP master problems. The 20 test problems in Table 2 involve a variety of applications. These include selection of configurations in chemical complexes, design and synthesis of batch processes, retrofits of heat exchanger networks and steam and power plants, complex distillation column designs and synthesis of process flowsheets. As can be seen, problems with up to 60 0-1 variables and 700 constraints and variables require modest computational effort (from few seconds to 2 minutes on an IBM-3090).

Additional computational experience with MINLP can also be found in Kocis and Grossmann (1989) and Sahinidis and Grossmann (1989b). The latter authors have reported results which to our knowledge correspond to the largest MINLP problem ever reported. The problem corresponds to an MINLP scheduling model for continuous parallel production lines involving 780 0-1 variables, 23,000 continuous variables and 3,200 constraints. GBD was used by exploiting the structure of the NLP subproblem whose dual solution can be obtained very efficiently, while the NLP subproblems reduce to small optimization problems when fixing the 0-1 variables. Convergence was achieved in fewer than 30 iterations requiring 20 minutes of CPU-time on an IBM-3090 computer.

Among the major trends that can be identified from the experience in solving MINLP problems, we can cite the following.

Firstly, problem formulation is one of the most crucial aspects for the successful solution of MINLP problems. Major features that can greatly enhance the efficient solution and convergence to the global optimum are tight NLP relaxations which can be accomplished for instance by tightly bounding the continuous variables, specifying smallest upper bounds in logical constraints such as in (6) and replacing weak integer constraints such as in (7) by a stronger set of inequalities. This has the effect of reducing the number of branches or major iterations in GBD or in the OA variants. To maximize the occurrence of linear constraints and minimize the occurrence of nonlinear functions is another desirable guideline for modelling, as this enhances the robustness of the solution of the NLP subproblems and tends to minimize the

effect of nonconvexities. Also, one should avoid if possible the use of products of 0-1 variables with continuous variables or functions as this often introduces nonconvexities. Lastly, one should try to reduce the number of 0-1 variables by exploiting the connectivity in a superstructure; this has the obvious effect of reducing the potential size of the tree that is to be examined by the branch and bound methods for MINLP and MILP.

Secondly, no algorithm is consistently superior in all the applications. Branch and bound is clearly superior if the relaxed NLP happens to exhibit integer solutions. As this is usually not the case, both GBD and the variants of the OA algorithm will normally outperform branch and bound which in large problems may require the solution of hundreds or thousands of NLP subproblems. As for the two latter algorithms, the family of OA algorithms will normally require much fewer major iterations (typically 3 to 5) than GBD, although the expense in solving the MILP master problem will be greater. For modest number of binary variables this is often not a serious limitation. However, this can become the major bottleneck in the computations if the relaxation in the MILP master is poor and the number of integer variables is large (e.g. see Yee and Grossmann, 1988).

Although the advantage in GBD is that the MILP master problem is easier to solve than in the OA methods, the number of major iterations with GBD can be somewhat unpredictable (many cases 5 to 20, but sometimes up to one hundred with many infeasible NLP subproblems). This can be a serious limitation if the NLP subproblems are expensive to solve. One way to reduce the number of major iterations in GBD is to either add extra constraints to the master problem and/or define some of the continuous variables as complicating variables for the master problem so as to hopefully strengthen the lower bound (e.g. see Yee and Grossmann, 1988; Sahinidis and Grossmann, 1989b; Ciric and Floudas, 1988b) which then however increases the expense in solving the MILP.

Thirdly, nonconvexities can cause difficulties in two ways. Firstly, the solution to the NLP subproblems may not be unique, and secondly the master problems in GBD and in the OA variants may cut-off the optimal solution. At present, a promising method to overcome nonconvexities for bilinear NLP subproblems is the partitioning scheme for GBD by Floudas *et al.* (1988), who have demonstrated the effectiveness of this scheme for the NLP optimization of the superstructure for heat exchanger networks (Ciric and Floudas, 1988a). As for the master problem, due to the strong convexity assumptions, the original OA and OA/ER algorithms are the most sensitive to nonconvexities, while GBD tends to be less affected by this problem. This is due to the fact that convexity is only required in the Lagrangian function, and that the master problem in GBD is poorly constrained and therefore has less likelihood of cutting-off the global optimum. On the other hand the most recent AP/OA/ER variant by Viswanathan and Grossmann (1989) has shown a remarkable degree of robustness to nonconvexities comparable to GBD. For instance all 20 problems in Table 2 converged to the global optimum despite the fact that half of these problems involve nonconvexities.

Finally, some other issues or observations that have emerged in the numerical solution are the following. A desirable feature in the OA algorithms is that the solution of NLP subproblems can be made successively easier to solve by using as a starting point the continuous variables predicted by the master problem (see Kocis and Grossmann, 1989a). This follows from the fact that as iterations proceed the master problem becomes an increasingly better approximation of the MINLP problem. This feature cannot be exploited in GBD since its master problem does not involve continuous variables. In the case of branch and bound one can also obtain good guesses for the NLP from the solution of the previous node, although if the relaxation gap is large, the quality of the guesses for the initial nodes will not always be very good. As for the solution of the MILP master problems, the requirements by the OA algorithms will be the highest. In fact, for larger problems we have found that unless one resorts to an advanced MILP package (e.g. MPSX, SCICONIQ these problems are prone to failure due to the accuracy that is required for the function linearizations, and to the more effective pruning techniques and features (e.g. special ordered sets) that are needed for large number of 0-1 variables. Future developments in cutting plane techniques (Crowder *et al.* 1983, Van Roy and Wolsey, 1987) may offer the possibility of solving with reasonable expense large MBLPs with poor relaxation. Alternatively, recent search techniques (e.g. tabu search, Glover (1988), simulated annealing (Aaarts and van Laarhoven, 1985), neural networks (Carpenter and Grossberg, 1988)) could be used instead of solving directly large MINLP problems with poor relaxation.

SOLUTION STRATEGIES FOR MINLP IN ENGINEERING SYSTEMS

One obvious approach to the MINLP optimization of engineering systems is to formulate the problem and solve it directly with any of the algorithms discussed in the previous section. A number of successful applications have been reported in the chemical engineering literature using such an approach (Kocis and Grossmann, 1987,1988; Foster, 1987; Floudas and Paules, 1988; Yee and Grossmann, 1989; Ciric and Floudas, 1988b; Duran and Flores, 1988; Kalitvenzeff and Marechal, 1989; Piboleau *et al,* 1989; Wellons and Reklaitis, 1989). In mechanical engineering Loh and Papalambros (1989) have reported several applications using a similar approach. However, it is clear that in order to increase the reliability and the efficiency of the solution procedure, one ought to recognize the special structure and properties that characterize the optimal synthesis of engineering systems (e.g. separable functions). Furthermore, there are potentially numerical difficulties when components "disappear" from a superstructure as this can lead to singularities or inconsistent linearizations.

Up to this date, not much work has been done in the development of solution strategies that are explicitly suited to engineering systems. Below we briefly describe a recent modelling-decomposition (M/D) strategy that has been proposed by Kocis and Grossmann (1989b) for the synthesis of process flowsheets and which is especially suitable for heterogeneous networks where there is a one-to-one correspondence between components and functions. Its major motivation has been to simplify the solution of the NLP and MILP problems, and to reduce the

undesirable effect of nonconvexities and of having to optimize "deleted components[1]" with zero flows which are temporarily turned off in the superstructure. The solution of the NLP is simplified by optimizing only the particular flowsheet at hand, as opposed to optimizing it within the superstructure as implied by problem (9). The MILP master problem is simplified by only incorporating at each iteration an approximation to the particular flowsheet. Finally, the effect of nonconvexities is reduced by special modelling techniques.

The basic idea in the M/D strategy is to first recognize that a flowsheet superstructure can be viewed as a network consisting of two types of components or nodes: interconnection nodes (splitters and mixers) and process unit nodes (reactors, separators). In summary, the modelling is performed as follows. Since interconnection nodes play a crucial role in defining the flowsheet structures and they exhibit well defined equations, special modelling techniques can be applied to these nodes. In particular, splitters and mixers that imply the choice of one single alternative can in fact be modelled through linear constraints which avoid the nonconvexities associated to the use of split fractions. For the case when multiple choices are possible, one can in fact develop valid linear outer-approximations that properly bound the nonconvex solution space in the MILP master problem. As for the process unit nodes, the mass balances are expressed in terms of component flows rather than in terms of fractional compositions. Lastly, the right hand side in the linearizations of the process units are modified to ensure that nonzero-flows are attained when the 0-1 variable is set to zero.

As for the decomposition part of this strategy the idea is as follows. Suppose we start by optimizing a particular flowsheet structure. It is clear that for the existing process units we are able to obtain linear approximations for the master problem. The question is then how to generate linear approximations of the "deleted" units in the superstructure. This can actually be accomplished by suboptimizing groups of units that are tied with existing interconnection nodes. Since prices (i.e. multipliers) and nonzero flows are available at these nodes, these can be used to suboptimize the nonexisting units "as if they were to exist" in the superstructure. This then provides not only nonzero flow conditions, but also points that are often very good for approximating these units. An example of how a superstructure based on an initial flowsheet can be decomposed into subsystems to be suboptimized is illustrated in Fig. 12 for the superstructure in Fig. 7. In this way, by optimizing the initial flowsheet structure, and suboptimizing the groups of nonexisting units, it then simply suffices to optimize the specific flowsheet that is generated at subsequent iterations in order to update the MILP. This then has two desirable effects: to only solve the NLP for each specific flowsheet, and to reduce the size of the MILP since only linearizations of existing units are incorporated at each iteration. This strategy is currently being automated in the flowsheet synthesizer PROSYN by Kravanja and Grossmann (1989).

Note that the strategy described above is still based on the idea of a simultaneous solution procedure where decomposition is being exploited by the problem structure. It is clearly conceptually sound to use a simultaneous approach since this reduces the risk of distorting trade-offs as interactions are explicitly accounted for (see also Grossmann, 1985, 1989). Decomposition schemes where the problem is sequentially partitioned according to a hierarchy of goals have shown to often, lead to suboptimal solutions*

APPLICATIONS

In order to illustrate more explicitly the capabilities of the MINLP approach to synthesis, three example problems will be presented.

Firstly, Yee and Grossmann (1989) have developed a new superstructure representation for heat exchanger networks (see Fig. 4). As shown in this figure a sequence of stages is postulated where in each stage all the potential matches between hot and cold streams are considered. The objective of the MINLP model is to determine a network configuration that minimizes the cost of the heat exchangers and the cost of the utility streams. This problem can be formulated as an MINLP problem in which the 0-1 variables are used to model the selection of each match which defines its corresponding heat exchanger, continuous variables are used to model the temperatures of the streams at each of the stages. Flow variables are not required since outlet temperatures for each stream exiting a given stage are assumed to be the same. Since the calculation of the areas can be substituted into the objective function the constraints in the MINLP all become linear, a highly desirable feature as discussed previously in the paper.

Consider the application of this MINLP problem to the stream data given in Table 3. This problem was first solved using two stages which led to an MINLP with 9 0-1 variables, 41 continuous variables and 62 constraints. The synthesized network which is shown in Fig. 13.a, was obtained with DICOPT++ in 4 major iterations and 12.5 sec on an IBM-3083. Although this network structure has minimum cost, it has the undesirable feature that it involves stream splits which could make it difficult to control. One can however, easily impose as a constraint that no stream splits be performed with the 0-1 variables (i.e. every stream can match at most one in each stage) This led to the network structure shown in Fig. 13.b. which is only 1% more expensive. Thus, this example shows that with an MINLP model the designer has considerable control on the structure that is to be synthesized. Also, as a point of interest, the designs that were obtained are cheaper by 10% when compared to the common heuristic methods that are available for heat exchanger network synthesis (e.g. see linnhoff and Hindmarsh, 1983). Also, as it turns out most of the heuristics are in fact violated at the optimal MINLP solution.

Secondly, Kocis and Grossmann (1989b) have recently synthesized the hydrodealkylation of toluene process for producing benzene. This problem has been studied extensively by Douglas (1985, 1988). The superstructure for this process (heterogeneous network with one-to-one relationship between components and functions) is shown in Fig. 14a which was derived based on a preliminary qualitative analysis of alternatives described in

Douglas (1988). Given the basic options considered for selection of reactors, use of membrane separators and absorbers, and a restricted set of alternatives for the separation and recycle, it is a relatively simple task to develop the superstructure representation. In this case the simplified nonlinear models were used to model the problem as an MINLP, which involves 13 0-1 variables, 672 continuous variables, and 678 constraints (140 nonlinear equations, 467 linear equations, 71 linear inequalities). The optimal solution, which is shown in Fig. 14b, was obtained with both the M/D strategy and with the AP/OA/ER algorithm on the full MINLP using MINOS and MPSX. The M/D strategy required 2 min of CPU time (IBM-3083), while AP/OA/ER required 8 min; both took 2 major iterations. This then shows the desirability of developing strategies that can exploit the structure of flowsheets. Furthermore, the example also shows how a qualitative pre-screening can lead to MINLP problems that are of reasonable size.

Lastly, Viswanathan and Grossmann (1989) have developed an MINLP model for determining the optimal feed tray location and the number of plates in distillation columns. The superstructure which is shown in Fig 15, consists of a number of potential trays, with subsets of them having potential feeds and by-pass streams to the condenser and reboiler. The mathematical model is rather complex as it involves rigorous material, heat balances, equilibrium equations and thermodynamic correlations. The specific example that was considered involved the separation of benzene and toluene and was modelled tray by tray with ideal thermodynamic correlations. For the case when a fixed number of 26 trays was specified, the MINLP involved 10 0-1 variables, 238 continuous variables and 239 constraints. The optimal feed tray location from among 10 candidate plates was determined by solving the relaxed NLP in 19 sec (IBM-3083). For the case when the number of trays was optimized for a fixed feed tray location from among 30 candidate trays, the MINLP involved 30 0-1 variables, 338 continuous variables and 467 constraints. In this case, the AP/OA/ER algorithm required 4 major iterations and 103 sec (IBM-3090). This example then shows that MINLP methods can be applied to complex process models.

CONCLUDING REMARKS

In this paper we have presented an overview of MINLP strategies and algorithms for the synthesis of engineering systems. From this review, it is clear that there has been considerable progress with this approach over the last few years. While heuristics have been extensively advocated due to the skepticism and little hope that there has been with the mathematical programming approach, we have now evolved to a state where the modelling and solution of large-scale MINLP problems for synthesis can no longer be regarded as a utopia. Results by our group at Carnegie Mellon, and by other researchers clearly have shown that much improved designs can be obtained with the use of MINLP models. It should be also noted that MINLP techniques are starting to be used in the chemical industry (e.g. Foster, 1987; Nath *et al*, 1986; Caracotsios and Petrellio, 1989).

This paper has emphasized applications in the chemical engineering domain, in part due to the background of the author, but also because it is in this domain where this approach has been applied and studied more extensively. There is reason to believe that a number of the ideas presented in this paper should be applicable to other domains. Our research work at the Engineering Design Research Center has indicated that a number of synthesis problems in civil and mechanical engineering share a similar structure as the MINLP problems in chemical engineering. For instance, nonlinear performance models and linear discrete variables. However, one important difference, particularly in mechanical engineering, is the one-to-many relationship between components and functions (see Rinderle et aL 1988), although these also arise in chemical engineering as discussed in this paper. On the other hand, in the electrical engineering domain (e.g. VLSI circuits) synthesis problems tend to have a much more specific structure than the general MINLP problem, making the application of specialized combinatorial optimization techniques more appropriate.

Finally, while one can expect that the scope of MINLP optimization for the synthesis of engineering systems will increase, it is also clear that a number of important challenges remain unsolved, and which most likely will be the subject of future work. Below we cite few major open questions:

1. How to systematically develop superstructures, particularly for heterogeneous systems?
2. What is the role of the new generation of combinatorial search techniques for synthesis such as neural networks, simulated annealing and tabu search?
3. How to effectively exploit the computational power of parallel computers to increase by several orders of magnitude the size of MINLP problems that can currently be solved?
4. How to effectively combine and integrate the MINLP optimization paradigm with qualitative AI techniques?

The latter question seems to be one of the most relevant ones since in principle both the optimization approach and the AI approach should complement the strengths of each other (e.g. see Glover, 1986; Simon, 1987).

ACKNOWLEDGMENT

The author would like to acknowledge financial support from the National Science Foundation under Grant CBT-8908735, and for support from the Engineering Design Research Center at Carnegie Mellon University. Access to the IBM-3090 computer at the Cornell Theory Center is also gratefully acknowledged.

Aarts, EJ1L- and van Laarhoven, P.J.M. (1985). Statistical Cooling: A General Approach to Combinatorial Optimization Problems. *Philips J. Res.* 40,193.

Andrecovich, MJ. and Westerberg, A.W. (1985). An MILP Formulation for Heat-Integrated Distillation Sequences Synthesis. *AIChE Journal* 31(9), 1461-1474.

Beale, E.M.L. (1977). Integer Programming. The State of the Art in Numerical Analysis (D. Jacobs, ed), Academic Press, London, pp 409-448.

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

Birewar, D.B. and Grossmann, I.E. (1989). Incorporating Scheduling in the Optimal Design of Multiproduct Batch Plants. *Computers and Chem. Eng.* 13, 141.

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

Carpenter, G. and Grossberg, S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network", *IEEE Computer* 21, 77-88.

Cavalier, T.M. and Soster, A.L. (1987). Logical Deduction Via Linear Programming. IMSE Working Paper 87-147, Department of Industrial and Management Systems Engineering, Pennsylvania State University.

Cerda, J. and Westerberg, A.W. (1983). Synthesizing Heat Exchanger Networks Having Restricted Stream/Stream Matches Using Transportation Problem Formulations, *Chemical Engineering Science* 38,1723-1740.

Ciric, A.R. and Floudas, CA.(1988a). Global Optimum Search for Heat Exchanger Network Optimization. *Proceeding ofPSE'88,* Sidney, 104-110.

Ciric, A.R. and Floudas, C.A. (1988b). Simultaneous Optimization of Process Stream Matches and Heat Exchanger Formulations. Paper No. 79e, Annual AIChE Meeting, Washington, DC.

Crowder, H., Johnson, E.L. and Padberg, M. (1983). Solving Large-Scale Zero-One Linear Programming Problems. *Operations Research* 31(5), 803-834.

Douglas, J.M. (1985). A Hierarchical Decision Procedure of Process Synthesis. *AIChE Journal* 31(3), 353-362.

Douglas, J. M. (1988). Conceptual Design of Chemical Processes, McGraw-Hill, New York.

Duran, M.A., Grossmann, LE. (1986a). A Mixed-Integer Nonlinear Programming Approach for Process Systems Synthesis. *AIChE Journal* **32(4),** 592-606.

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

Duran, M.A. and Grossmann, I.E. (1986c). Simultaneous Optimization and Heat Integration of Chemical Processes" *AIChE Journal* 1,123-138.

Duran, M.A. and Flores, A. (1988). Mixed-Integer Nonlinear and Linear Programming Approaches to the Synthesis of Heat Integrated Distillation Sequences. Paper No. 81c, Annual AIChE Meeting, Washington, D.C.

Floudas, C.A., Ciric, A.R. and Grossmann, I.E. (1986). Automatic Synthesis of Heat Exchanger Networks. *AIChE Journal* 32(2), 276-290.

Floudas, C.A. and Paules, G.E. (1988). A Mixed-Integer Nonlinear Programming Formulation for the Synthesis of Heat Integrated Distillation Sequences. *Computers and Chem. Eng.* **12(6),** 531-546.

Floudas, C.A., Aggarwal, A. and Ciric, A.R. (1988). Global Optimum Search for Nonconvcx NLP and MINLP Problems. Paper No. 76c, Annual AIChE Meeting, Washington, DC.

Garfinkel, R. S., Nemhauser, G. L. (1972). Integer Programming. John Wiley and Sons, New York.

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

Glover, F. (1986). Future Paths for Integer Programming and links to Artificial Intelligence. ***Computers and Optns. Res.*** **13, 533-549.**

Glover, F. (1988). Tabu Search. Center for Applied Artificial Intelligence, CAAI Report 88-3, Boulder, Colorado.

Goldfarb, D. and M.J. Todd (1989). Linear Programming. Optimization (eds. G.L. Nemhauser, A.H.G. Rinnoy Kan and M.J. Todd), North Holland, Amsterdam.

Grossmann, I.E. (1985). Mixed-Integer Programming Approach for the Synthesis of Integrated Process Flowsheets. *Computers and Chemical Engineering* **9**(5), 463-482.

Gundersen, T. and Naess, L. (1988). The Synthesis of Cost Optimal Heat Exchanger Networks-An Industrial Review of the State of the Art. *Computers and Chem. Eng.* **12**, 503-530.

Gupta, O.K. (1980). Branch and Bound Experiments in Nonlinear Integer Programming. Ph.D. Thesis, Purdue University.

Gupta, O.K. and Ravindran, A. (1983). Nonlinear Integer Programming and Discrete Optimization, *ASME Journal of Mechanism, Transmission and Automation in Design*, Vol. 105, 160-164.

Hendry, J.E. and Hughes, R.R. (1972). Generating Separation Process Flowsheets. *Chem. Eng. Progress* **68**, 69.

Hoffman, K. and Padberg, M. (1986). LP-Based Combinatorial Problem Solving. *Annals of Operations Research*, **4**, 145-193.

Hohmann, E.C. (1971). Optimum Networks for Heat Exchange. Ph.D. Thesis, University of Sothern California.

IBM Mathematical Programming System Extended/370 (MPSX/370), Basic Reference Manual, White Plains, NY (1979).

Karmarkar, N. (1984). A New Polynomial-Time Algorithm for Linear Programming. *Combinatorica*, **4**(4), 373-395.

Kocis, G.R. 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. (1988). Global Optimization of Nonconvex MINLP Problems in Process Synthesis. *Industrial and Engineering Chemistry Research* **27**, 1421.

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

Kocis, G.R. and Grossmann, I.E. (1989b). A Modelling/Decomposition Strategy for MINLP Optimization of Process Flowsheets. *Computers and Chem. Eng.* 13,797-819.

Kravanja, Z. and I.E. Grossmann, "PROSYN - An MINLP Process Synthesizer", manuscript in preparation (1989).

Lien, K, Suzuki, G., and Westerberg, A. W., "The Role of Expert Systems Techniques in Design" *Chemical Engineering Science,* 42(5), 1049-1071 (1987).

Linnhoff, B. and Hindmarsh, E. (1983). The Pinch Design Method of Heat Exchanger Networks. *Chemical Engineering Science* 38 745.

Loh, H.T. and Papalambros, P.Y. (1989). A Sequential Linearization Approach for Solving Mixed-Discrete Nonlinear Design Optimization Problems. Technical Report, Design Laboratory, University of Michigan.

Loh, H.T. and Papalambros, P.Y. (1989). Computational Implementation and Tests of a Sequential Linearization Algorithm for Mixed-Discrete Nonlinear Design Optimization. Technical Report, Design Laboratory, University of Michigan.

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

Maher, M.L. (1988). Engineering Design Synthesis: A Domain Independent Representation, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 1 (3).

Marsten, R., "Users Manual for ZOOM/XMP", The Department of Management Information Systems, University of Arizona (1986).

Mawengkang, R and Murtagh, B.A. (1986). Solving Nonlinear Integer Programs with Large-Scale Optimization Software. *Annals of Operations Research,* 5, 425-437.

Mawengkang, H. (1988). Nonlinear Integer Programming. Ph.D. thesis, The University of New South Wales, Australia.

Murtagh, B.A. and Saundcrs, M.A. (1985). MINOS User's Guide. Systems Optimization Laboratory, Department of Operations Research, Stanford University, "SOL 83-20".

Ncmhauscr, G.L. and Wolsey, L.A. (1988). Integer and Combinatorial Optimization. Wiley-Interscience, New York.

Papoulias, S. and Grossmann, LE. (1983). A Structural Optimization Approach in Process Synthesis, Parts I, n, and HI. *Computers and Chem. Eng.* 7(6), 695-734.

Papalambros, P.Y. and Wilde, P.J. (1988). Principles of Optimal Design: Modelling and Computation, Cambridge University Press, Cambridge.

Paules, G.E. and Floudas, C.A. (1987). APROS: A Discrete-Continuous Optimizer for Solution of Mixed-Integer Nonlinear Programming Problems. Presented at ORSA/TEMS Meeting, St. Louis, MO.

Pekny, J. and Miller, D. (1989). A Parallel Branch and Bound for Solving Large Asymmetric Traveling Salesman Problems. Presented at CORS/ORSA/ITMS Meeting, Vancouver.

Rinderle, J.R., Colburn, E.R., Hoover, S.P., Paz-Soldan, J.P. and Watton, J.D. (1988). Form-Function Characteristics of Electromechanical Designs. *Design Theory 88,* Springer Verlag.

Sahinidis, N. and Grossmann, LE. (1989a). Reformulation of the Multiperiod MILP Model for Capacity Expansion of Chemical Proccesses. Submitted to *Operations Research.*

Sahinidis, N. and Grossmann, LE. (1989b). MINLP Model for Scheduling in Continuous Parallel Production Lines. Presented at AIChE Meeting, San Francisco.

Sargent, R.W.H. and Gaminibandara, K. (1976). Optimal Design of Plate Distillation Columns. *Optimization in Action* (L.W.C. Dixon, ed.), 267-314, Academic Press London.

Schrijver,A. (1986) Theory of Linear and Integer Programming.John Wiley, New York.

Simon, H.A. (1987). Two Heads Are Better Than One: The Collaboration Between AI and OR. *Interfaces* 17,8-15.

Stephanopoulos, G. (1981). Synthesis of Process Flowsheets: An Adventure in Heuristic Design or a Utopia of Mathematical Programming?. *Foundations of Computer-Aided Chemical Process Design* (R.S.H. Mah and W.D. Seider, eds.), Engineering Foundation, New York, Vol. 2, pp 439-499.

Vasclcnak, J.A., Grossmann, LE. and Westerberg, A.W. (1987). Optimal Retrofit Design of Multiproduct Batch Plants. ***Industrial and Engineering Chemistry Research*** **26,**718-726.

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

Viswanathan, J. and Grossmann, LE. (1989). A Combined Penalty Function and Outer-Approximation Method for MINLP Optimization. Presented at CORS/TTMS/ORSA Meeting, Vancouver.

WellonsJLS. and Reklaitis, G.V. (1989). The Design of Multi-Product Batch Plants Under Uncertainty with Staged Expansions. ***Computers and Chem. Eng.*** **13(1/2), 115-126.**

Williams, H.P. (1978). The Reformulation of Two Mixed Integer Programming Problems. *Mathematical Programming* 14, 325-331.

Yee, T.F. and Grossmann, LE. (1988). A Screening and Optimization Approach for the Retrofit of Heat Exchanger Networks. Paper 8 Id, Annual AIChE Meeting, Washington, DC.

Yee, T.F. and Grossmann, LE. (1988). A Simultaneous Optimization Approach for the Synthesis of Heat Exchanger Networks. Paper 136f, Annual AIChE Meeting, San Francisco.

Yuan, X., Zhang, S., Pibouleau, L. and Domenech, S. (1987). Une Methode d'optimisation non lineaire en variables mixtes pour la conception de procedes. Partie I: Presentation de l'agorithine. Submitted for publication as the RAIRO Recherche Operationnellc.

Table 1

Comparison between GBD and the OA method on convex problems

Problem	Size			GBD	GBD	OA	OA
	0-1 Var	Cont. Var	Constr.	Iterations	CPU-time	Iterations	CPU-time
HW74	3	8	8	4	4.3	2	2.8
EX3	8	25	31	8	14.9	3	10.8
BATCH2	9	12	19	4	6.2	3	7.1
BATCH8	40	38	142	66	2527	4	291

Notes:
1. Cpu-times in sec (VAX-8860)
2. NLP problems: MINOS; MILP problems: ZOOM

TABLE 2. Computational Results with DICOPT++

Problem	0-1 Var.	Cont. Var.	Constr.	Nonzeroes (nonlinear)	Iterations[1]	Time[2] (sec)	% NLP:MILP
LAZIMY	2	8	5	22(5)	1	0.06	100:00
HW74	3	9	9	28(2)	4	0.33	45:55
NONCON	3	3	6	17(2)	1	0.03	100:00
YUAN	4	4	10	32(9)	3	0.35	66:34
CAPITAL	10	3	7	46(2)	4	0.35	48:52
FLEX	4	12	16	47(4)	3	0.37	67:33
REL1	16	21	18	69(36)	3	3.77	52:48
EX3	8	26	32	101(5)	5	0.82	51:49
EX4	25	7	31	227(127)	5	12.33	12:88
BATCH	24	23	74	191(22)	3	1.67	92:08
BATCH8	40	33	142	353(32)	4	5.06	76:24
BATCH 12	60	41	218	545(40)	4	9.96	52:48
TABATCH	24	71	129	462(124)	3	4.23	68:32
UTILRED	28	118	168	467(10)	3	8.2	19:81
TFYHEN	30	74	144	465(18)	3	22.7	20:80
EX5FEED	10	238	239	1103(826)	1	5.4	100:00
UN15FEED	12	587	586	3318(2336)	1	66.8	100.00
EX5T11	30	338	467	1943(1278)	4	114.1	43:57
EX5T12	30	338	467	1943 (1278)	3	53.48	79.21
HDASS	13	709	719	2204(462)	4	123.9	77:23

*N iterations require N NLP suhproblems and N-1 MILP master problems.

[2]Touil CPU time, NLP:MINOS 5.2/MILP: MPSX 1.7, IBM-3090.

Table 3

Data for Heat Exchanger Network

Stream	TIN (C)	TOUT(C)	Fcp (kW/C)	Cost ($/kW-yr)
H1	443	333	30	-
H2	423	303	15	-
C1	293	408	20	-
C2	353	413	40	-
S1	450	450	-	80
W1	293	313	-	20

U - 0.8 (kW/m^2 C) for all matches except ones involving steam

U « 1.2 (kW/m^2 C) for matches involving steam

Annual Cost - 1000 * (Area(m^2))$^{0.6}$ for all exchangers except heaters

Annual Cost - 1200 * (Area(m^2))$^{0.6}$ for heaters

Captions of Figures

Fig. 1. Tree representation tor the separation of 4 chemical products.

Fig. 2. Network representation tor the separation of 4 chemical products.

Fig. 3. Network representation tor alternative designs in a structure.

Fig. 4. Heat exchanger network superstructure by Yee et al. (1989).

Fig. 5. Heat exchanger network superstructure with one-to-many relationships by Yee et al. (1988).

Fig. 6. Superstructure by Sargent and Gaminibandara (1976) tor the separation of 4 products.

Fig. 7. Flowsheet superstructure tor specific alternatives.

Fig. 8. Plot of feasible region tor alternative constraint models.

Fig. 9. Superstructure for the synthesis of a chemical industrial complex.

Fig. 10. Branch and bound enumeration tor a problem with three 0-1 variables.

Fig. 11. Main steps in the GBD and OA methods.

Fig. 12. Decomposition of flowsheet superstructure in Fig. 7.

Fig. 13. Heat exchanger network designs tor the data in Table 3.

Fig. 14. Superstructure and optimal flowsheet of HDA toluene process.

Fig. 15. Superstructure tor feed tray location and number of plates in a distillation column.

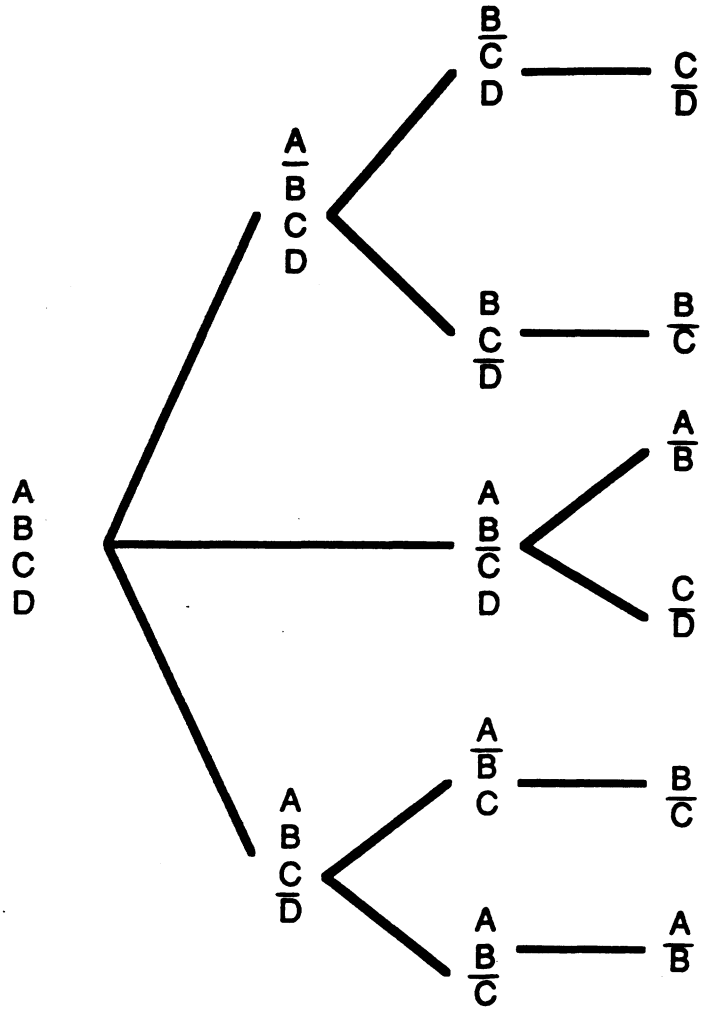


Fig. 1

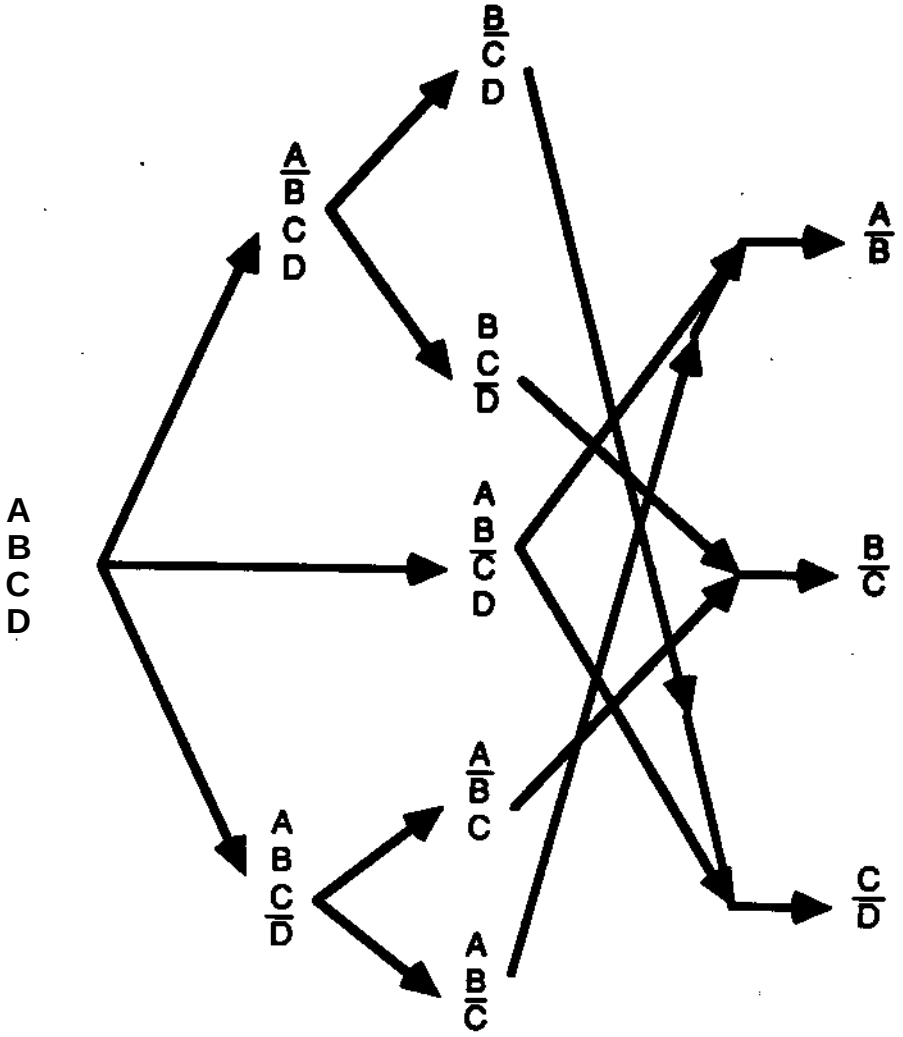
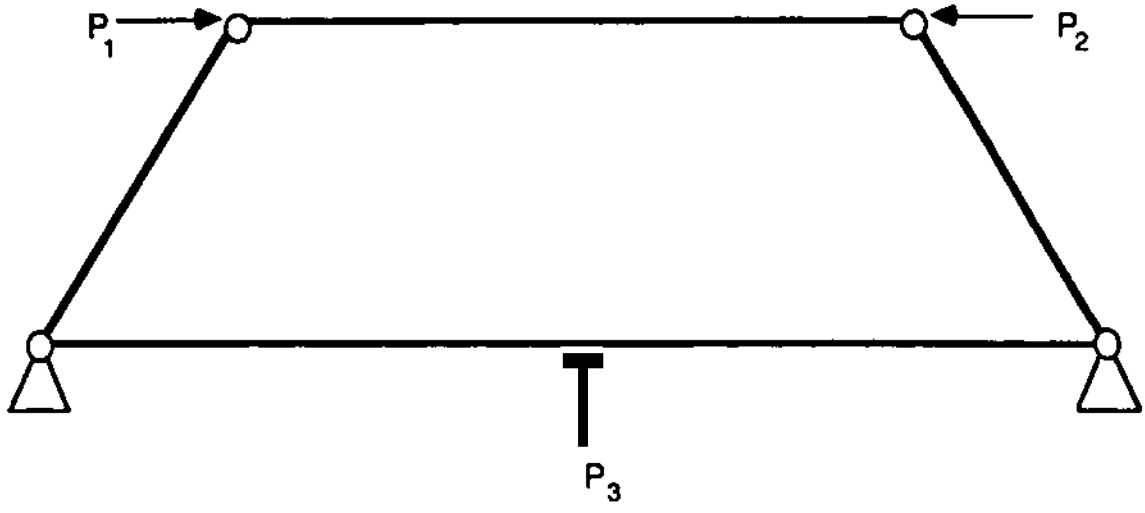
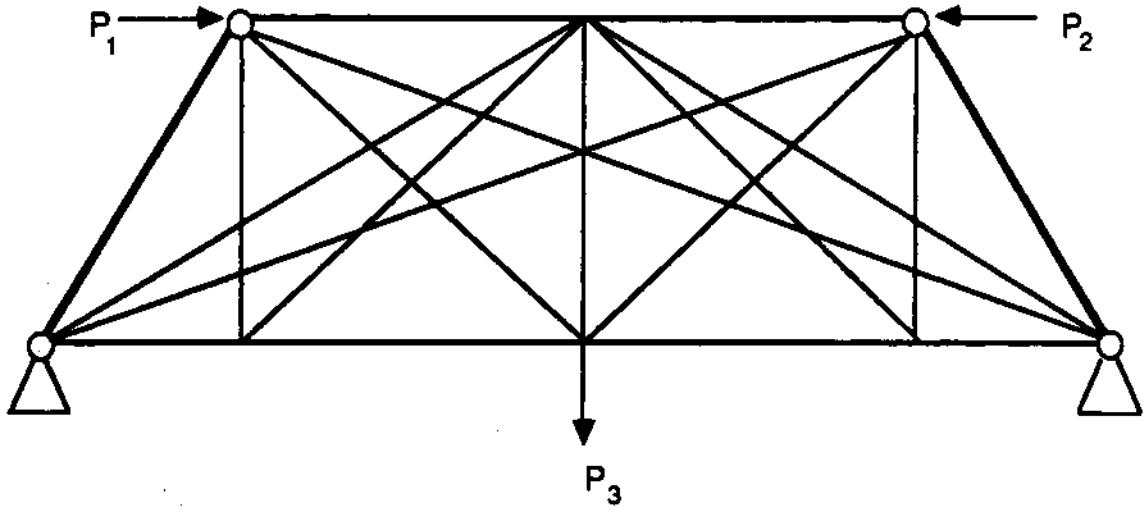


Fig. 2



(a) Structure involving three loads.



(b) Superstructure containing alternative selections of bars.

Fig. 3

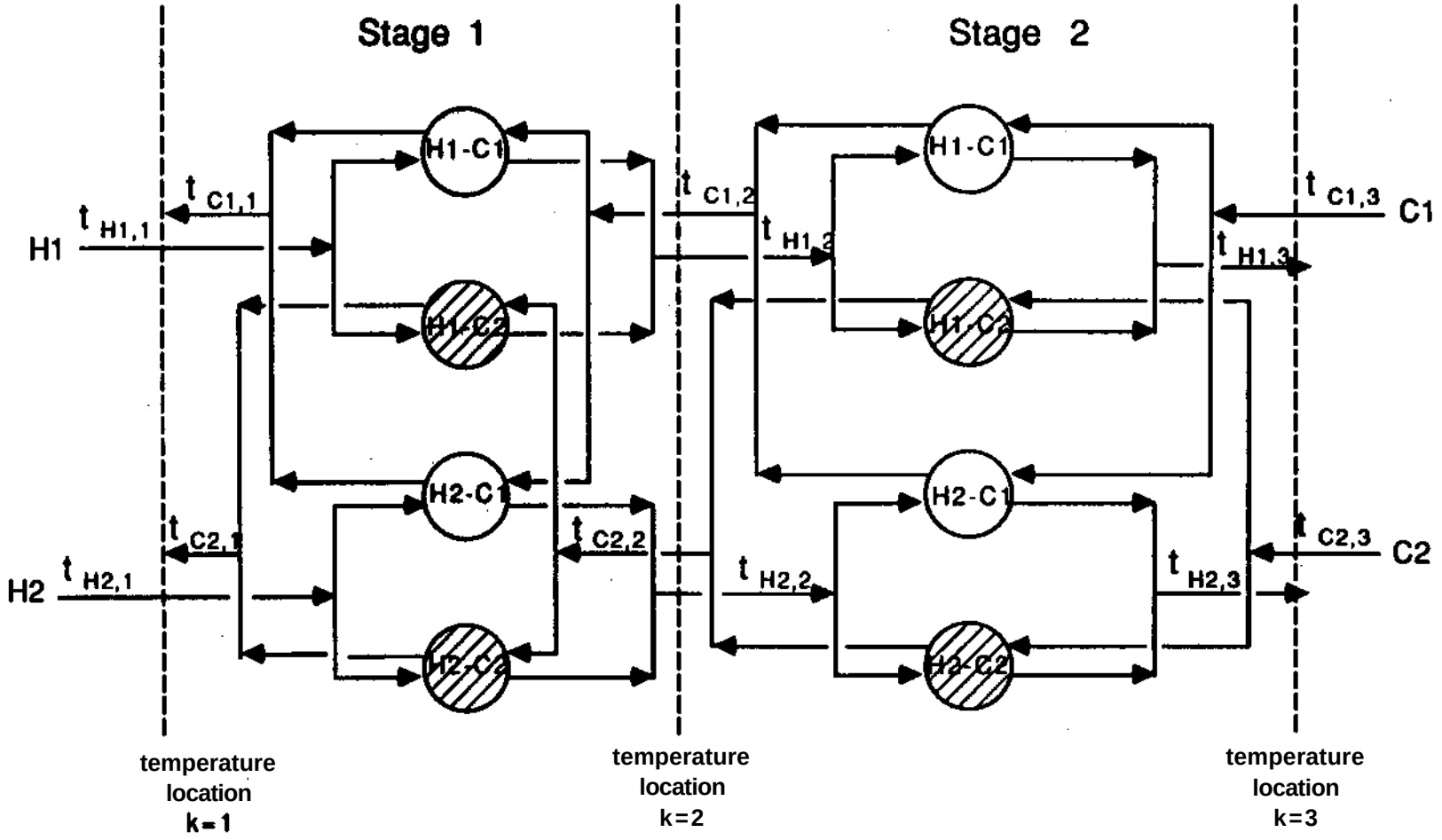

Stage 1
Stage 2
H1-C1
H1-C2
H2-C1
H2-C2
H1
H2
C1
C2
t H1,1
t C1,1
t C1,2
t H1,2
t C1,3
t H1,3
t C2,1
t H2,1
t C2,2
t H2,2
t C2,3
t H2,3
temperature location k=1
temperature location k=2
temperature location k=3

Fig. 4

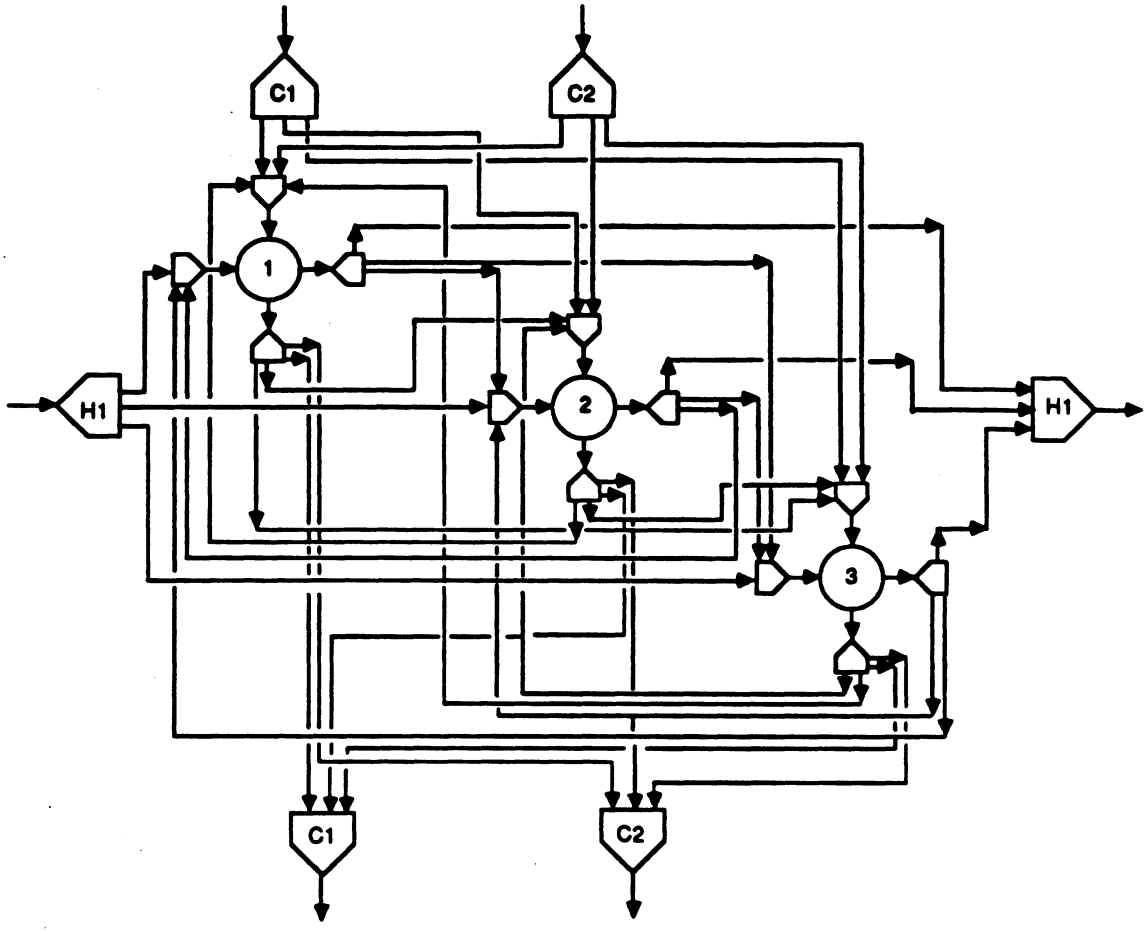


Fig. 5

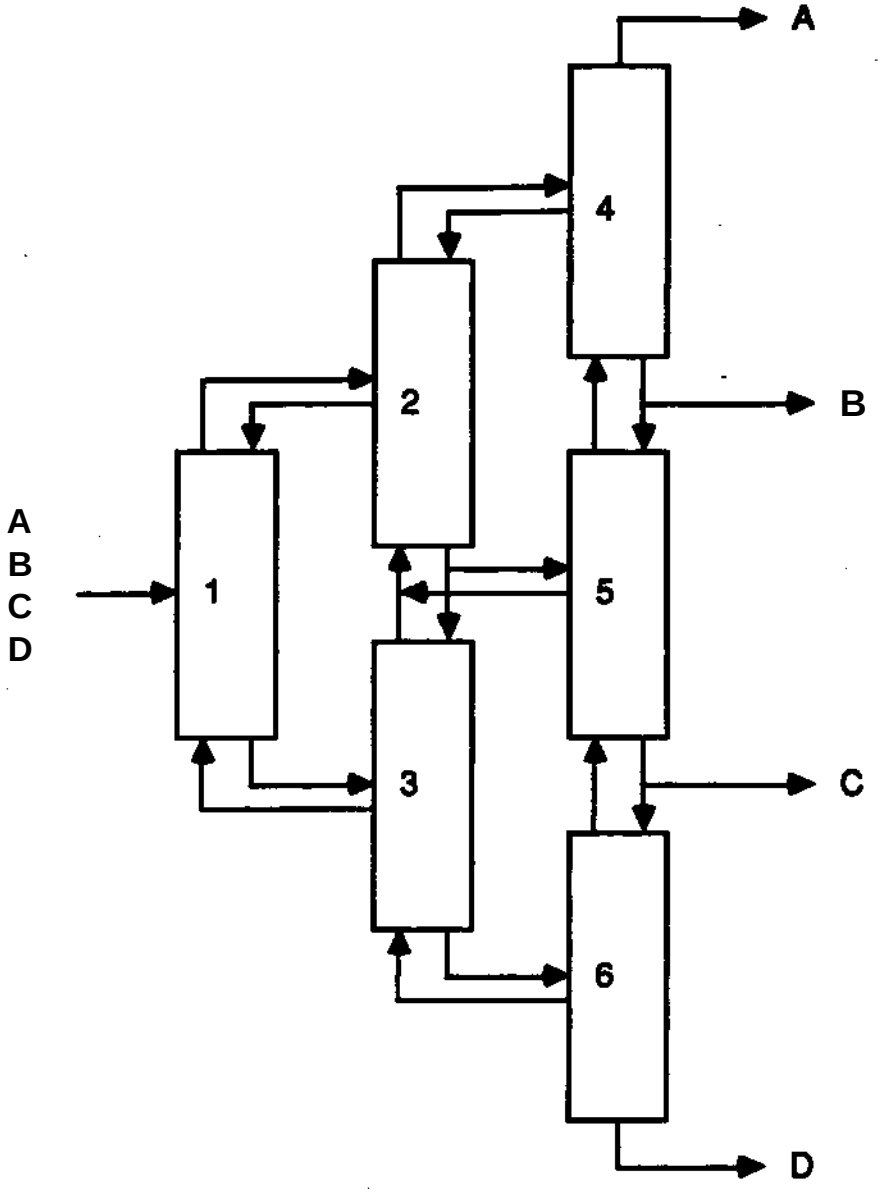
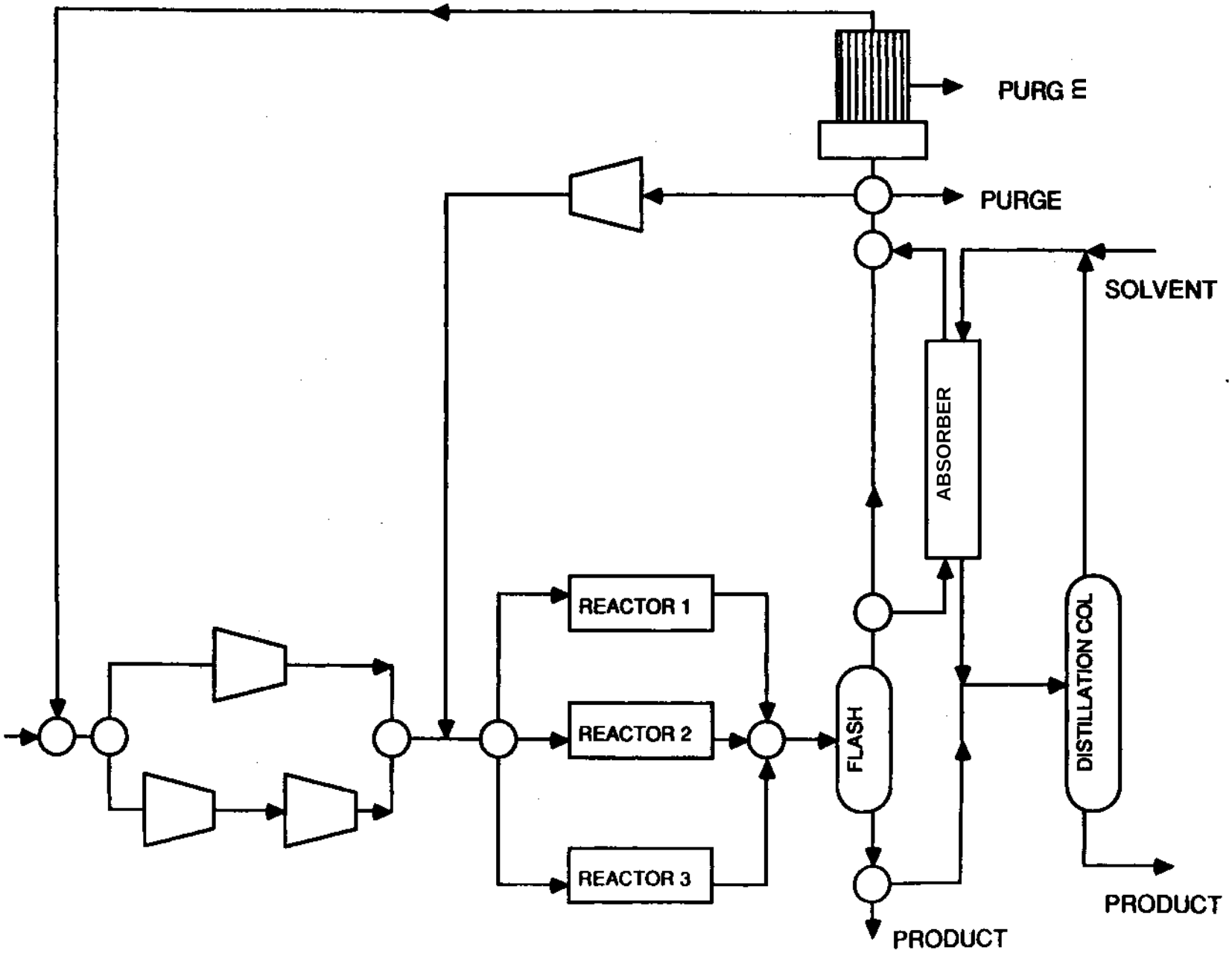
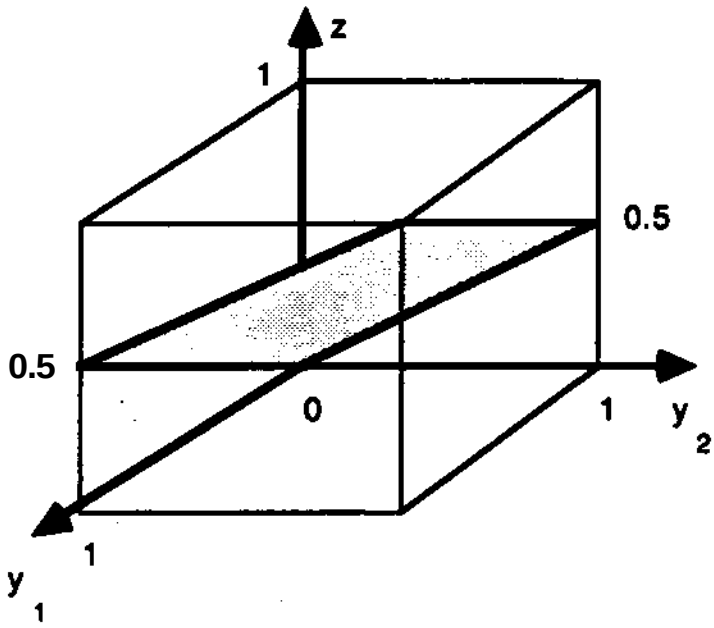


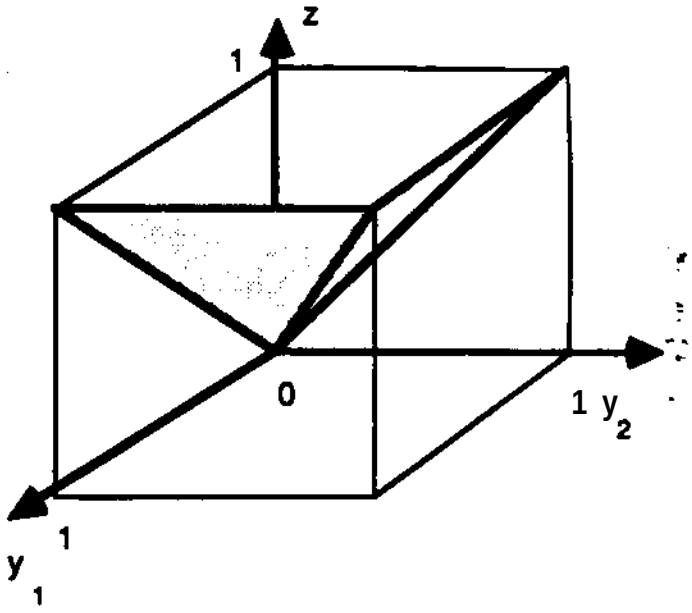
Fig. 6


PURG E
PURGE
SOLVENT
ABSORBER
REACTOR 1
REACTOR 2
REACTOR 3
FLASH
DISTILLATION COL
PRODUCT
PRODUCT

Fig



(a) $y_1 + y_2 - 2z \leq 0$



(b) $y_i - z \leq 0$

$y_2 - z \leq 0$

Fig.8

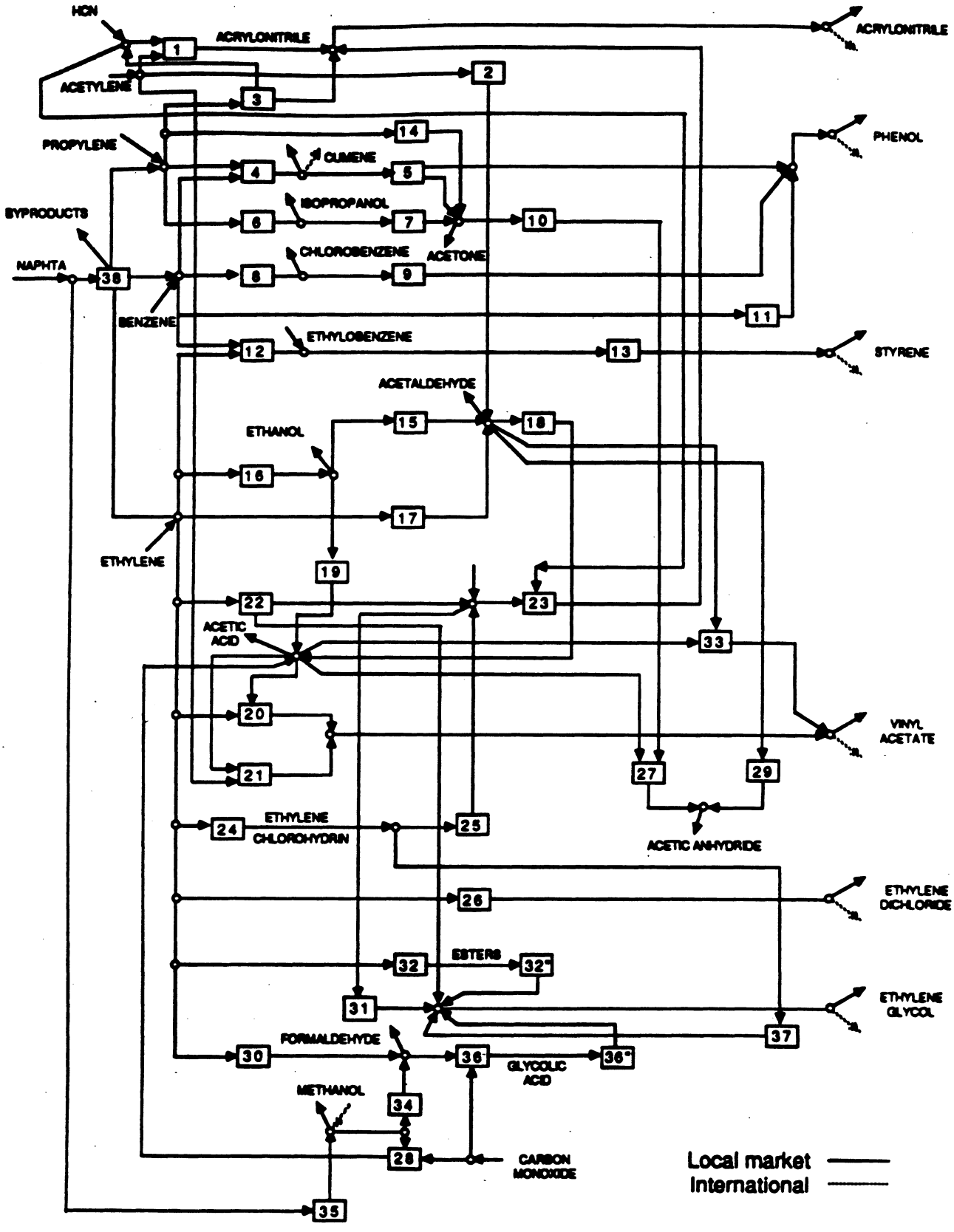

HCN
ACRYLONITRILE
ACETYLENE
PROPYLENE
CUMENE
PHENOL
ISOPROPANOL
BYPRODUCTS
CHLOROBENZENE
ACETONE
NAPHTA
BENZENE
ETHYLOBENZENE
STYRENE
ACETALDEHYDE
ETHANOL
ETHYLENE
ACETIC ACID
VINYL ACETATE
ACETIC ANHYDRIDE
ETHYLENE CHLOROHYDRIN
ETHYLENE DICHLORIDE
ESTERS
ETHYLENE GLYCOL
FORMALDEHYDE
GLYCOLIC ACID
METHANOL
CARBON MONOXIDE
Local market
International

Fig. 9

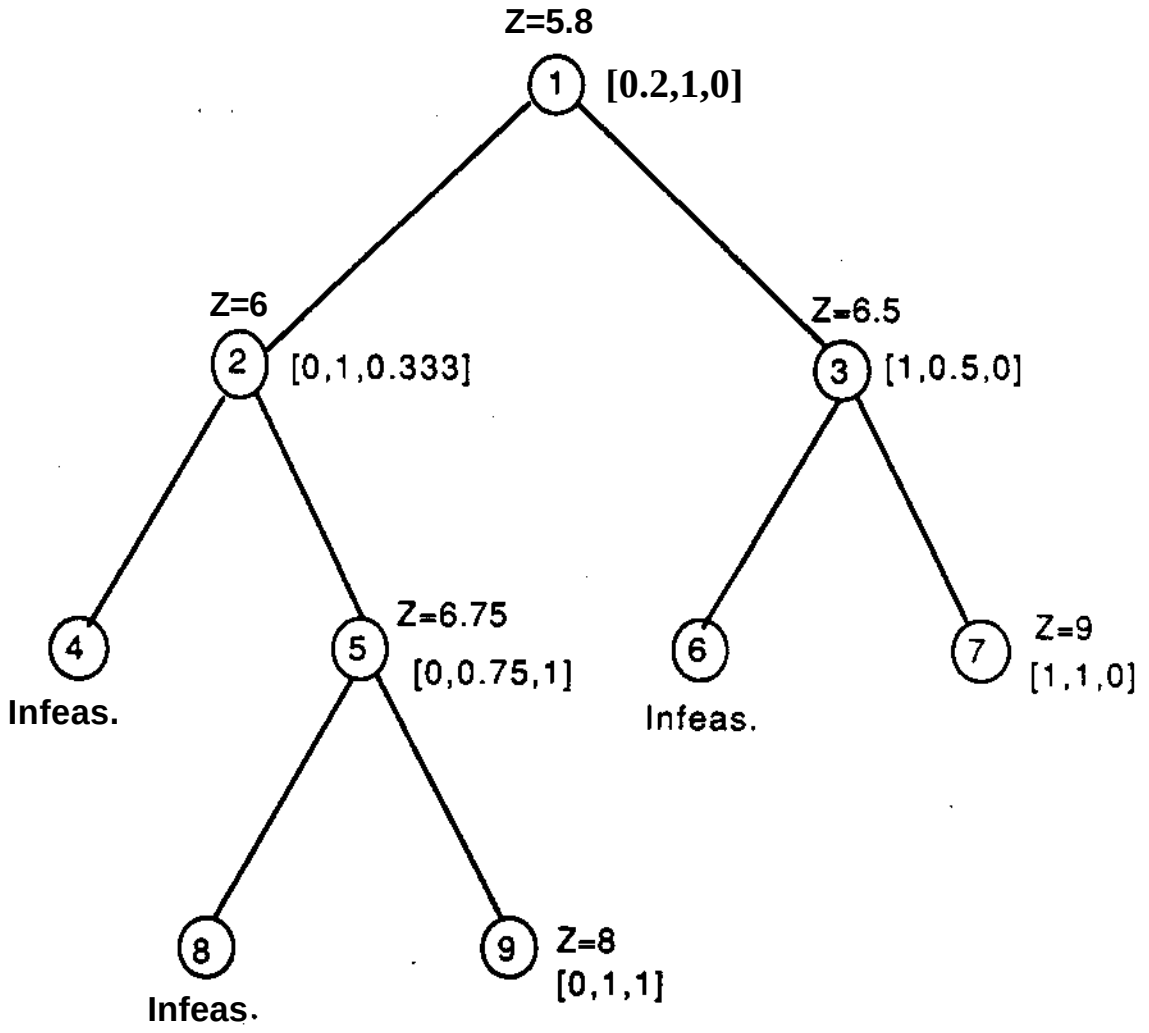

Z=5.8
1
[0.2,1,0]
Z=6
2
[0,1,0.333]
Z=6.5
3
[1,0.5,0]
4
Infeas.
5
Z=6.75
[0,0.75,1]
6
Infeas.
7
Z=9
[1,1,0]
8
Infeas.
9
Z=8
[0,1,1]

Fig. 10

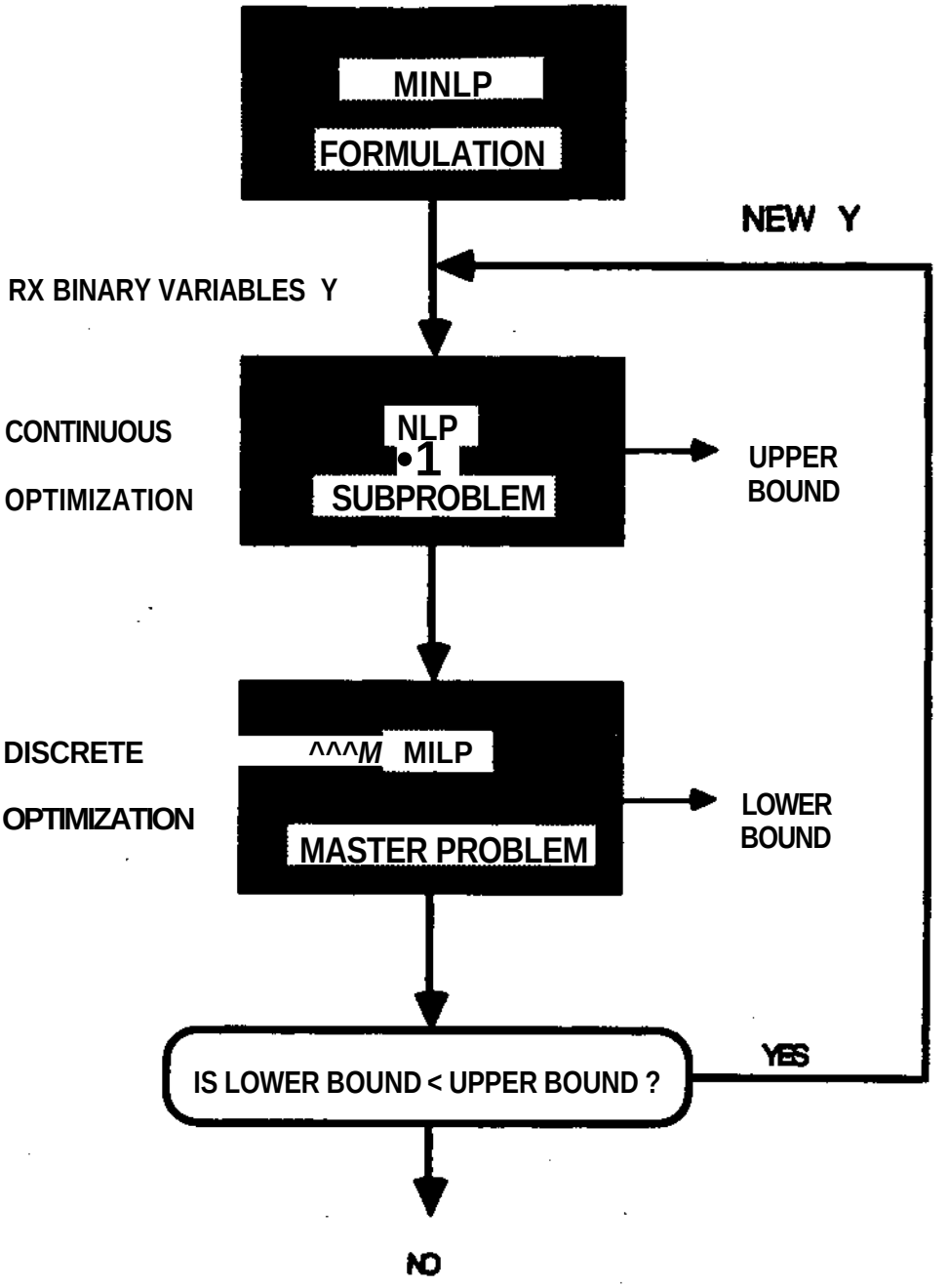
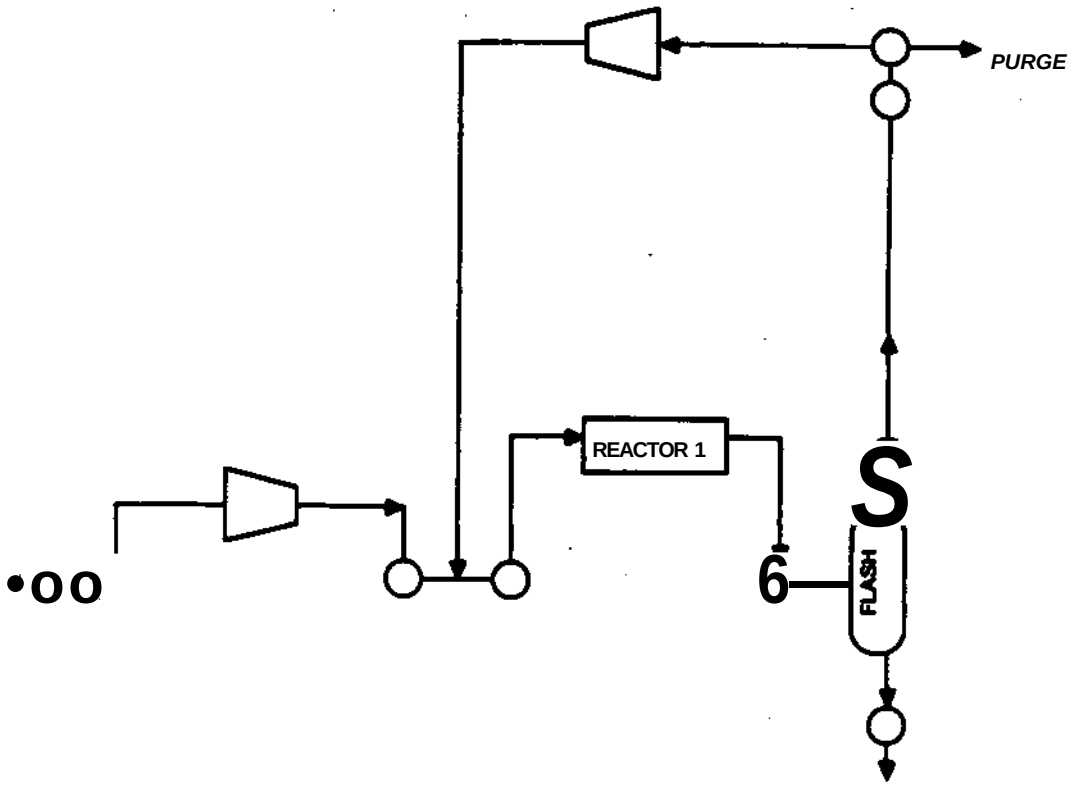
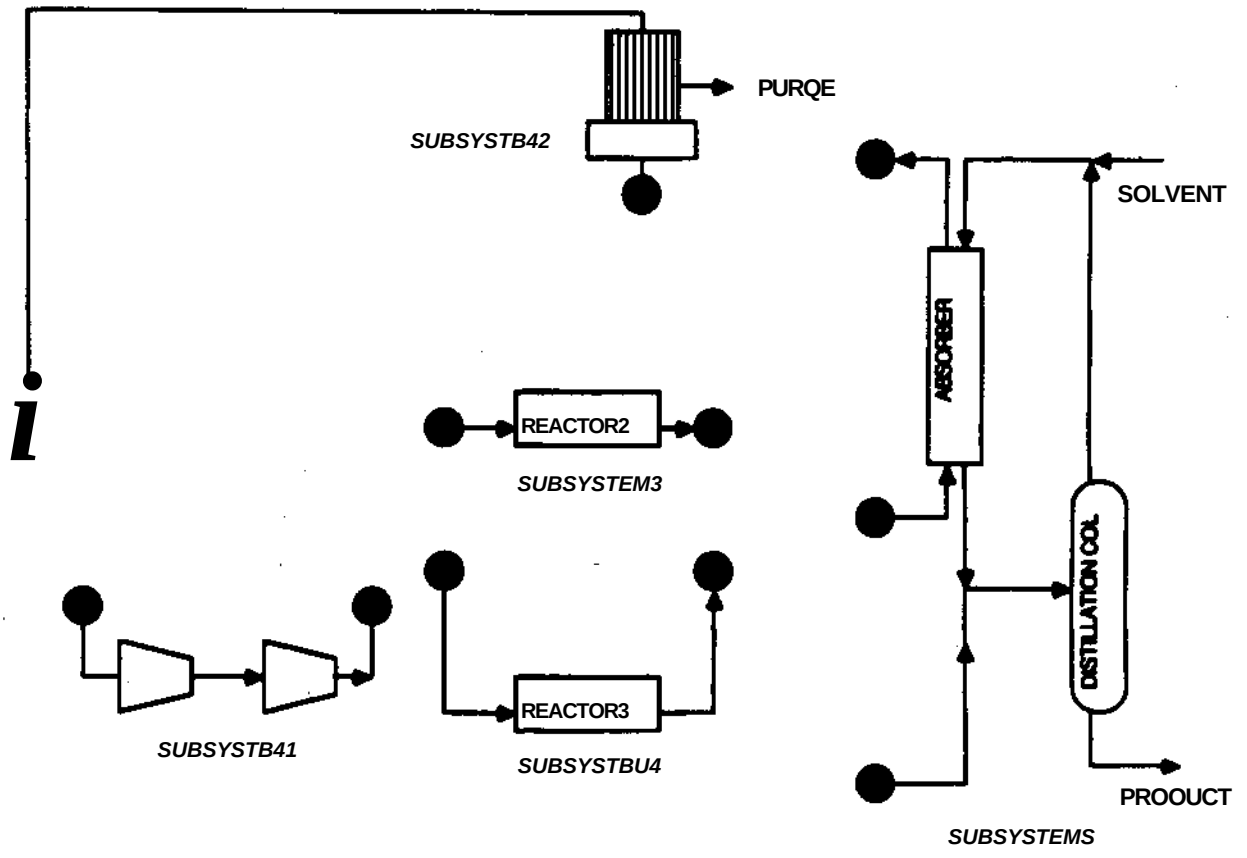

MINLP
FORMULATION
NEW Y
RX BINARY VARIABLES Y
CONTINUOUS
OPTIMIZATION
NLP
•1
SUBPROBLEM
UPPER
BOUND
DISCRETE
OPTIMIZATION
^^^M MILP
MASTER PROBLEM
LOWER
BOUND
IS LOWER BOUND < UPPER BOUND ?
YES
NO

Fig. 11

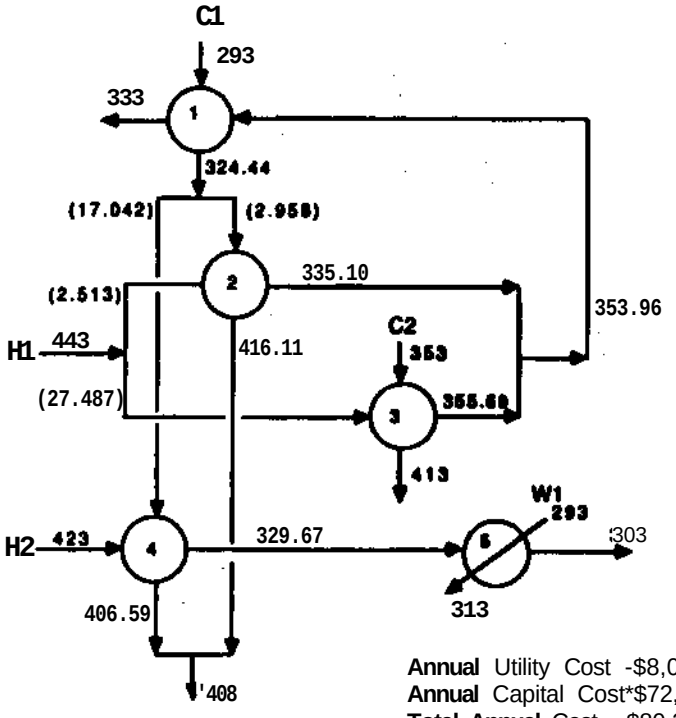


(a) Initial Flowsheet



(b) Subsystems for suboptimization

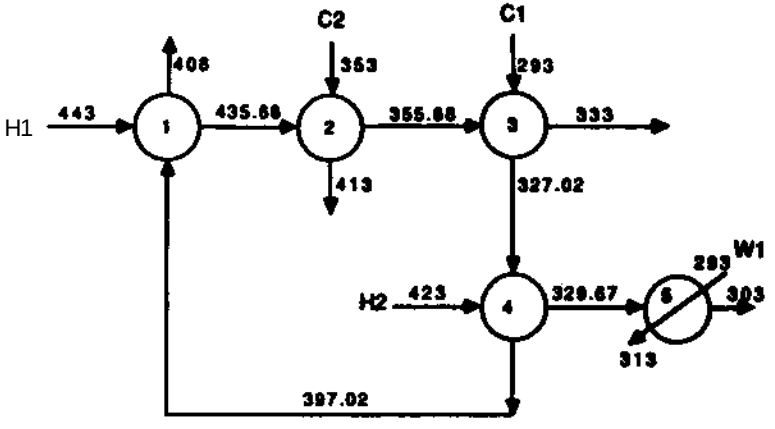
Fig. 12



Annual Utility Cost -$8,000
Annual Capital Cost*$72,274
Total Annual Cost -$80,274

Each.	HttfLoad	Altai*^{2})
1	626.6	22.6
2	271.2	19.3
3	2400	265.1
4	1400	179.0
5	400	36.3

(a) No constraints on splits

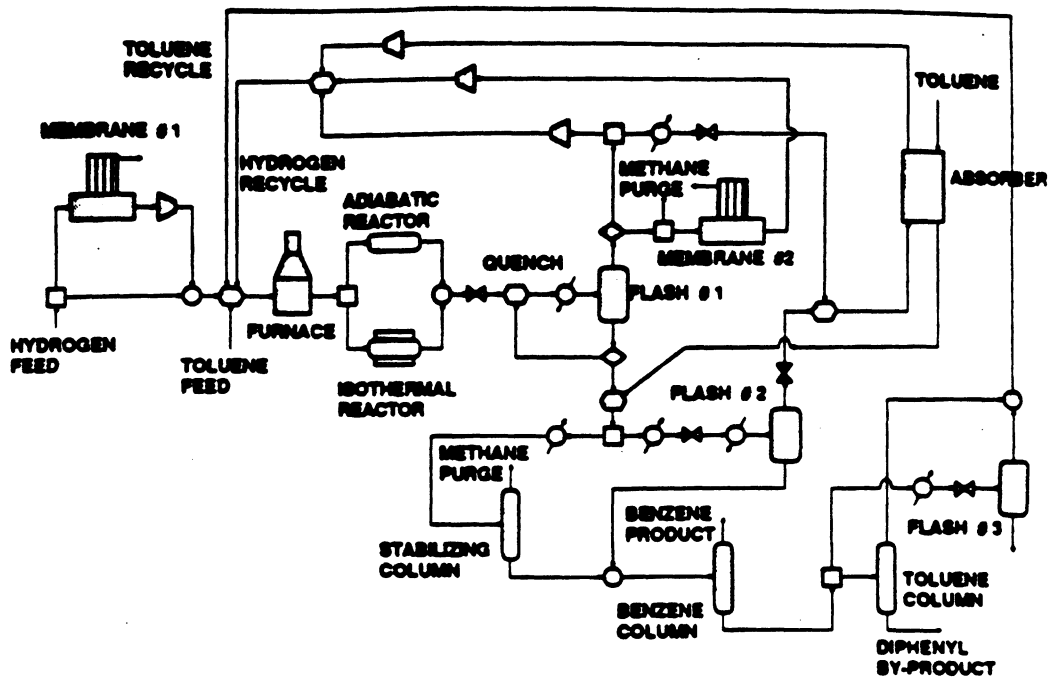


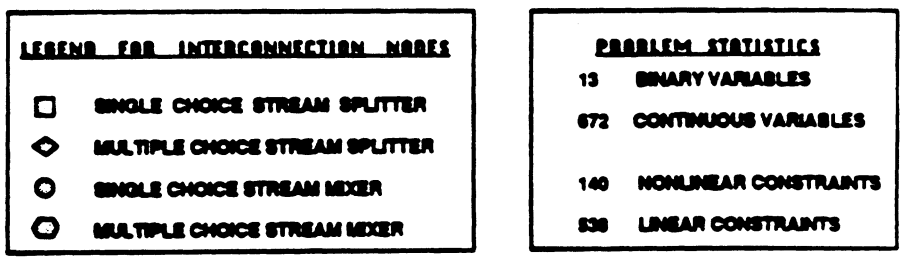
Annual Utility Cost - $8000
Annual Capital Cost - $72,909
Total Annual **Cost** - **$80,909**

Exch.	HMILMd (kW)	Aitafln*)
1	219.6	7.5
2	2400	320.3
3	680.4	25.0
4	1400	171.3
5	400	36.3

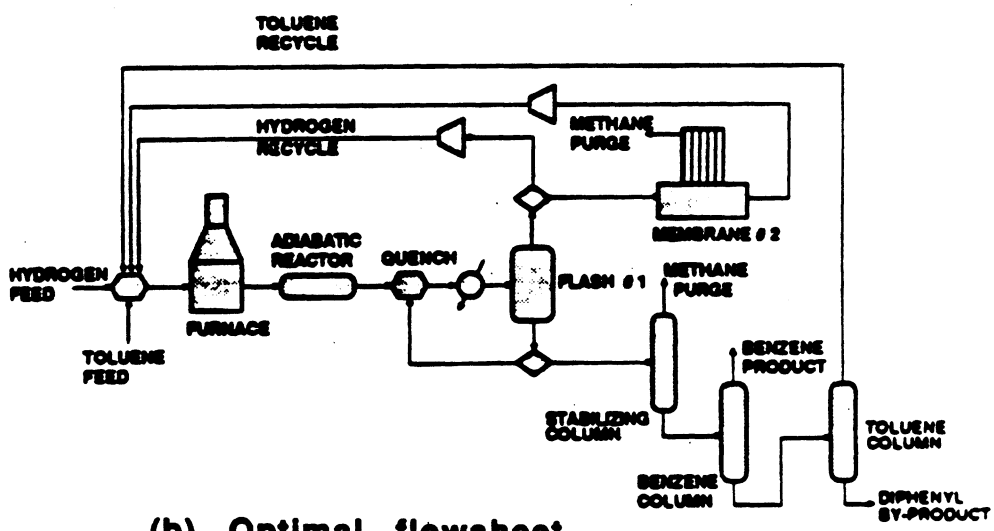
(b) No splits allowed

Fig. 13





(a) Superstructure



(b) Optimal flowsheet

Fig. 14

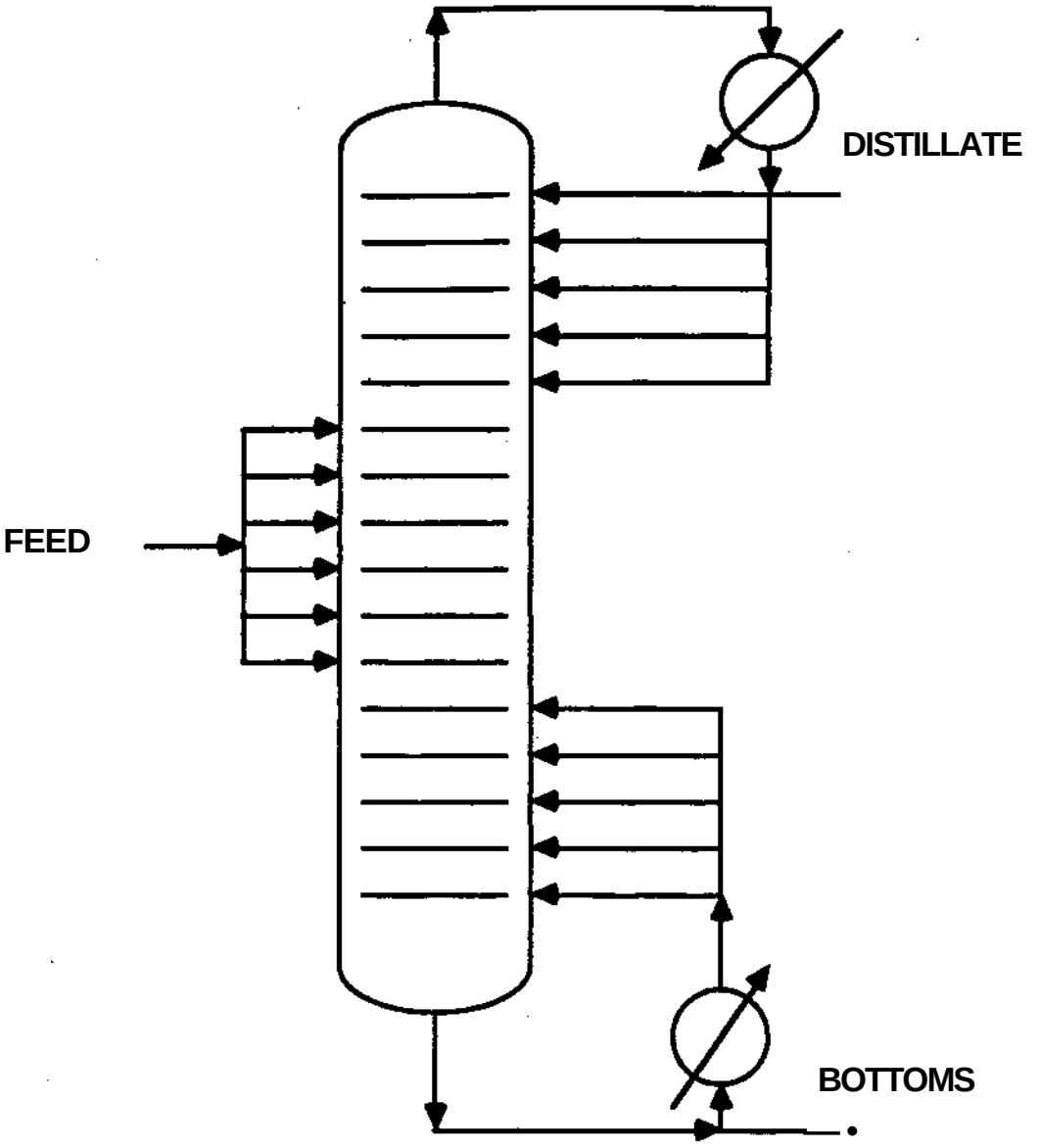


Fig. 15