1992

Logic cuts for processing networks with fixed charges

John N. Hooker
Carnegie Mellon University

Carnegie Mellon University.Engineering Design Research Center.

Logic Cuts for Processing Networks with Fixed Charges

J.N. Hooker, H. Yan, I.E. Grossmann, R. Raman

EDRC 06-127-92

Logic Cuts for Processing Networks with Fixed Charges *

J. N. Hooker, H. Yan, I. E. Grossmann, R. Raman
Engineering Design Research Center
Carnegie Mellon University, Pittsburgh, PA 15213 USA

August 1991
Revised April 1992

Abstract

We show how some simple logical constraints can substantially accelerate the solution of mixed integer linear programming (MILP) models for the design of chemical processing networks. These constraints are easily generated in a preprocessing stage and can be applied either symbolically during a branch-and-bound search or as constraints in the MILP model. Furthermore, they represent a new class of cuts, "logic cuts," that generalize traditional cutting planes. A logic cut can cut off feasible points but does not change the optimal solution. We establish some elementary properties of logic cuts and use them to show that our logical constraints for processing networks exhaust all possible logic cuts for these problems. Preliminary computational results are presented, using OSL.

1 Introduction

Mixed integer programming models arise in many practical applications (e.g. see Williams [13]). Unfortunately, the solution of these models can be quite expensive with the LP-based branch and bound methods that are

***The authors gratefully acknowledge financial support from the Engineering Design Research Center at Carnegie Mellon University. The Center is funded by NSF grant No. 1-55093. The first author is also partially supported by AFOSR grant 91-0287.**

implemented in many commercial and academic computer codes (e.g. OSL, MPSX, SCICONIC, ZOOM, LINDO, APEX, etc.). This has motivated in recent years substantial research work for improving the computational efficiency of MILP algorithms. These include the development of reformulation techniques and cutting plane algorithms, as for instance in the work by Van Roy and Wolsey [12], Sherali and Adams [11] and Balas, Ceria and Conniejols [2]. However, these new developments have concentrated on numerical aspects, mainly on convexification and reduction of the integrality gap so as to minimize enumeration with branch and bound.

Although these numerically based techniques are quite promising, an alternate and complementary direction is to exploit the logical structure of an MILP problem (see Jeroslow [8]). A problem may have logical constraints that restrict the number of solutions that need to be enumerated, although these constraints may not be explicit in the MILP model. Constraints of this kind, once they are identified, can be used either as additional inequality constraints within the MILP model or as symbolic constraints that restrict the generation of alternatives in a branch-and-bound search.

This paper shows how some simple logical constraints can be used to accelerate the solution of chemical processing network problems with fixed costs. The object in these problems is to design a network of processing units to yield desired outputs as profitably as possible. A network containing more units than necessary is given, and integer variables are used to indicate which units are actually installed. Each unit employed incurs a fixed cost.

In a problem of this kind it obviously makes no sense to install a unit if nothing flows through it. We would therefore like to add constraints that prevent a unit from being installed if it carries no flow, so as to reduce the number of alternatives examined. But no such constraint can be represented in an MILP, because it would result in a feasible region that is not a closed set. We can, however, include logical constraints that prevent the installation of a unit that *cannot* carry flow, either because upstream units from which it could receive flow have not been installed, or similarly for downstream units. These constraints can have also the converse effect of forcing the installation of downstream or upstream units if the unit in question is installed.

Although it is a simple matter to identify constraints of this sort as part of a preprocessing stage, they are not identified by any preprocessor of which we are aware. In fact, we found that we can solve even fairly small problems

substantially faster than a state-of-the-art MILP code with a preprocessor (OSL).

We believe that the constraints we identify are interesting not only because they help solve processing network problems, but also because they represent a class of cuts that apparently has not been recognized before and that could prove useful for other problems. Unlike traditional valid cuts, they can cut off feasible solutions of the problem, although they do not change the value of the optimal solution. We therefore provide a rigorous definition for these cuts, which we call *logic cuts,* and establish some of their elementary properties. These properties can be used to determine whether one has identified all possible logic cuts for a given problem, and we illustrate this for processing networks.

Whereas a cut in the traditional sense is an inequality, a logic cut can take the form of any restriction on the possible values of the integer variables, whether or not it is expressed as an inequality. Two logic cuts, however differently expressed, are the same logic cut if they exclude the same values. Thus logic cuts can be used to prune a search tree even when they are not expressed as inequality constraints in an MILP model. But they can also be imposed as inequalities within an MILP model, in which case they can tighten the linear relaxation and cut off fractional solutions as traditional cuts do.

We will define a logic cut to be a constraint on the values of the integer variables that does not change the projection of the problem's epigraph onto the space of continuous variables. Furthermore, a logic cut must have this property for any set of objective function coefficients, provided the integer variables have nonnegative coefficients. Logic cuts therefore cut off integer points that are dominated by others, in a sense we make precise below.

This definition is partially motivated by the work of R. Jeroslow [8], who viewed integer variables as artificial variables used solely to define the shape of the epigraph in continuous space. From this perspective it is natural to admit cuts that leave the problem in continuous space undisturbed even if they cut off feasible solutions in the original space.

We begin in Section 2 with a small illustration of logic cuts, define them rigorously in Section 3, and prove some elementary lemmas in Section 4 that characterize them. In Section 5 we illustrate the use of logic cuts in a small processing network problem. In Section 6 we provide, for a class of generalized processing networks, a characterization of logic cuts that is

complete in the sense that the cuts we describe imply all (nonvalid) logic cuts for the problem. These cuts are easily generated in polynomial time. In Section 7 we solve several processing network problems to show the effect that logic cuts have in reducing the number of nodes in the branch and bound tree. We also make a comparison between putting the logic cuts in the MILP and handling them symbolically. The final section suggests some avenues for further research.

2 Illustration of a Logic Cut

To illustrate the idea of a logic cut, we consider a very simple optimization problem on the network of Fig. 1. A flow of at most M is available at node 1, nodes 3 and 4 are sinks, and node 2 conserves flow. The flows on arcs (1,2) and (2,4) are $x\backslash$ and x_2, and they generate revenues $c\backslash X\backslash$ and C2X2, respectively, where Cj is any real number. But arcs (1,2) and (2,4) must be constructed at a cost of $d\backslash$ (> 0) and d_2 (> 0), respectively, before they can carry flow. This suggests the MILP,

$$\begin{aligned} \max \quad & c_x x\backslash + c_2 x_2 - d_x y_x - d_2 y_2/2 \qquad (1) \\ \text{s.t.} \quad & x_2 \leq \pounds 1 \\ & xj \leq Myj, \quad j = 1,2, \\ & *;\geq 0, \quad yj \in \{O,l\}, \quad j = 1,2. \end{aligned}$$

Obviously there is no point in constructing either arc unless it carries flow. Yet nothing in the constraint set rules out solutions with (XJ,VJ) = (0,1). Nor can we rule out all such solutions with additional MILP constraints. This is because a set 5 of all points feasible for (1) except those with $(xjjijj)$ = (0,1) fails to contain some of its limit points and thus is not a closed set. For instance, points with (xi,yi) = (0,1) are limit points because S contains points arbitrarily close to them. But the feasible set for any MILP must be a finite union of polyhedra (as pointed out by Jeroslow [7]) and therefore must be closed. This means that S is the feasible set of no MILP.

We can, however, add constraints that rule out *some* of these spurious solutions. We can observe that the constraint $x\backslash \leq My\backslash$ forces xi to zero when y_i = 0. But $x\backslash$ = 0 implies x_2 = 0, in which case we can suppose y_2 = 0. We conclude that we can force y_2 = 0 whenever yi = 0. It is easy

to write a constraint for *this* relation, since it is a logical relation between two y/s:

$$Vi\text{-}y2 > \_\ 0. \tag{2}$$

If we regard *yj* as a logical proposition that is true when *yj* = 1 in (1) and false when *yj* = 0, (2) can also be written as a logical formula:

$$\text{Vi V -i}y_2, \tag{3}$$

where V means "or" and -» means "not." Constraints (2) and (3) axe alternate ways of formulating the same *logic cut* They do not exclude all solutions in which (x_2, VT) = (0* 1)* but they rule out (x_2, y_2) = (0,1) when yi = 0 as well.

Also note that the above logic cut has the effect of forcing *y* to a value of one if y_2 is fixed to one. If such a cut is not present, then depending on the value of M, t/i may take on a fractional value when y_2 is set to one.

Although an existing preprocessor may conceivably generate cut (3) as an implication y_2 => *Vi* in this simple problem, no preprocessor of which we are aware can generate the logic cuts we identify for processing networks.

3 Definition of a Logic Cut

We now use the concepts of an epigraph and its projection to provide a rigorous definition of a logic cut. Consider a mixed integer programming problem,

$$\begin{aligned} \max \quad & ex - dy \\ \text{s.t.} \quad & Ax + By \leq a \\ & Xj \geq 0, \ j = 1,\ldots,n, \\ & yj \in \{0,1\}, \ j = 1,\ldots,p, \end{aligned} \tag{4}$$

where *A* is m x n, *B* is m x p, and each *dj* ≥ 0. (If *dj* < 0, replace yy with 1 - *yj.)* The objective function can be viewed as a function defined on the domain *D* described by the constraints. The *graph G* of this function, which we refer to as the graph of (4), is the set {(*,y, *)!(£, y) € -D, *z* = *ex* - *dy}.* The *epigraph E* is everything "on or below" the graph, namely {(x,y,z)|(x,y) *e D, z* ≤ *ex* — *dy}.* The optimal value of the optimization problem (4) can be written max{z|(x,y,z) G *E}.*

The graph of (1) is the union of the following sets:

$$\{(0,0,0,0,0)\}$$
$$\{(x_1, 0, 1, 0, c_1x_1 - \text{rfi}) | 0 \leq \text{xj} < M\}$$
$$\{(0, 0, 0, 1, -d_2)\}$$
$$\{(x_1, x_2, 1, 1, c_1x_1 + C_2X_2 - di - <\text{f}_2) | 0 \leq X_2 \leq \text{Xx} \leq M\}$$

The *projected graph Gp* is the projection of *G* onto the space of continuous variables, so that $Gp = \{(x,z)|(x,y,z) \in G\}$. The *projected epigraph* is $\text{£p} = \{(x,z)|(x,y,z) \in E\}$. Clearly the optimal value of (4) is $\max\{z|(x,z) \in Ep\}$. This means that we can remove points from the epigraph *E* without changing the value of the optimal solution, provided the projected epigraph *Ep* does not change.

The projected graph for (1) is the union of the following sets.

$$\{(0,0,0)\}$$
$$\{(\text{xj}, 0, c_x x_x - \text{rfi}) | 0 \leq \text{X!} \leq \text{Af}\}$$
$$\{(0,0,-*)\} \qquad (6)$$
$$\{(x_1, x_2, c_1x_1 + C_2X_2 - d_X - <\text{f}_2) | 0 \leq X_2 \leq \text{X!} \leq M\}$$

This set is depicted by the heavy points and line segments and the shaded triangle in Fig. 2. The projected epigraph is the union of this set with all the points below it.

A *logic cut* for (4) is a restriction on the possible values of *y* that, when applied, has no effect on the projected epigraph *Ep* of (4). Thus we have,

lemma 1 *The addition of a logic cut to the constraint set of (4) does not change the value of the optimal solution.*

For example, the cut (3) removes the point $(0,0,0,\text{l},-<\text{f}_2)$ from the graph of (1). Thus it removes from the projected graph *Gp* only the point $(0,0,-\text{d}_2)$ (Fig. 2). The projected epigraph *Ep* is unchanged, since $<\text{f}_2 \geq 0$ implies that the origin lies directly above this point. (3) is therefore a logic cut and does not affect the optimal solution value.

4 Characterizing Logic Cuts

We can usefully characterize logic cuts as follows. If *D* is the feasble set for (4), let us say that a point $y \in \{0,1\}^P$ is *feasible* if $(x,y) \text{£} D$ for some x.

Also a point $y' \in \{0,1\}^P$ is *dominated* by y if $y \leq y^1$ and $(x, y) \in D$ whenever $(*,*) \in D$.

Lemma 2 $7/5 \subset \{0,1\}^n$,

$$y \in 5 \tag{7}$$

is a logic cut for (4) if and only if every feasible $j/ \pounds S$ *is dominated by some feasible* $y \in S$.

Proof Suppose first that (7) is a logic cut, and let $y^7 \pounds 5$ be feasible. Then removing any point of the form (x, y^7) from D does not change Ep. This means that the projection $(x, ex - dy^*)$ of any $(x, y, ex - dx/) \in G$ lies below the projection $(x, ex - dy)$ of some point $(x, y, ex - dy) \in (7$ for which $y \in S$. Thus $ex - d\backslash f \leq ex - dy$. Since $if \geq 0$, $y \leq y'$, and it follows that $j/$ is dominated by y.

Conversely, suppose every feasible $t/ \wedge S$ *is* dominated by some feasible $y \in 5$. Then since $d \geq 0$, the projection $(x,cx - dy^*)$ of any $\{x \wedge yf \wedge cx - dxf) \in G$ lies below the projection $(x, ex - dy)$ of $(x, y, ex - dy)$. It follows that (7) is a logic cut. a

In the above example, we can note that $(xi,X2_10,1) \in D$ only if $(xi,X2,-0,0) \in D$. Thus $y' \$ \{(0,0), (1,0), (1,1)\}$ is feasible only if it is dominated by a point $y \in \{(0,0), (1,0), (1,1)\}$, namely (0,0). We therefore have a logic cut,

$$y \in \{(0,0), (1,0), (1,1)\}, \tag{8}$$

.which is equivalent to (3).

If S contains all feasible points, the logic cut (7) is a valid cut. Cut (8) is nonvalid because S does not contain the feasible point (0,1).

In pure integer programming (i.e., when the variables x do not appear in (4)), y dominates y' if and only if $y \leq y'$. Lemma 2 therefore becomes very simple.

Corollary 1 *(7) is a logic cut for a pure integer programming problem if and only if for every feasible* $y' \pounds 5$, *there is some feasible* $y \in S$ *for which* $y \leq y'$.

We now address the problem of determining whether all nonvalid logic cuts have been identified. In the above example, it is dear from (5) and (6) that no feasible values of *y* other than $(0_9 1)$ can be cut off without changing the epigraph. For instance, if we cut off (1,0), then we remove the set {(xi,0,Ci£i — <fi)|0_≤ *x*\ _≤ *M}* from *Gp* (represented by a line segment in Fig. 2). This clearly alters the projected epigraph *Ep.*

In general we can check whether we have found all nonvalid logic cuts by adding the known logic cuts to the constraint set and applying the following corollary to determine whether any more nonvalid logic cuts exist.

Corollary 2 *Problem (4) has a nonvalid logic cut (7) if and only if some feasible* $y^7 \in \{0,1\}^P$ *is dominated by some feasible* $y \neq y'$.

If we add logic cut (8) to (1), then no feasible *t/* is dominated by another feasible point. The only feasible points *y'* are (0,0), (1,0) and (1,1). When X! > 0, (xi,0,l,0) € *D* but (xi,0,0,0) g *D.* When $x_u x_2 > 0$, (si*£2> 1,1) € *D* but (xi,X2;l,0),(xi,X2,0,0) £ *D.* So, no more feasible points can be cut off.

5 Illustrative Example

We now illustrate how logic cuts can help solve a processing network design problem. Consider the simple processing network example in Fig. 3. There are three potential processes that can be selected for manufacturing *jd>* maximum of 10 tons/hr of chemical C. The maximum capacities of these processes are 30, 30 and 50 tons/hr respectively. In addition, there is a specification that processes 1 and 2 cannot be selected simultaneously.

The MILP model for the optimal selection of processes that minimizes the total cost is assumed to be given as follows:

$$\begin{aligned} \max \quad & Z = -\text{Hyx} - 12y_2 - 10y_3 - 3x_3 - 2.8x_5 + 9x_7 - 2x_x \qquad (9) \\ \text{s.t.} \quad & \text{xi} - x_2 - x_4 = 0 \\ & x_3 - 0.9x_2 = 0 \\ & x_5 - 0.85x_4 = 0 \\ & *6 - x_3 - x_5 = 0 \\ & x_7 - 0.75x_6 = 0 \end{aligned}$$

$$x_3 - 30y_1 \leq 0$$
$$*5 - 30y_2 \leq 0$$
$$x_7 - 50y_3 \leq 0$$
$$y_1 + y_2 \leq 1$$
$$*1,\ldots,*7 \geq 0;\ x_7 \leq 10;\ y_1, y_2, y_3 \in \{0,1\}$$

If this problem is solved using a standard branch and bound method, the enumeration tree that results is shown in Fig. 4. A total of 8 nodes must be examined. One could of course improve the formulation of the MILP model given above. For instance, the value of the upper bounds (maximimum capacities 30,30,50) of the logical constraints could be reduced to yield a tighter LP relaxation. While schemes such as this one will undoubtedly help the branch and bound method, the underlying problem is that solutions with zero flows and active processes are enumerated as seen in Fig. 4. Specifically, node 5 has process 1 active and zero flows, while node 7 has process 2 active and also zero flows. Furthermore, nodes with fractional solutions (nodes 2 and 4) may be enumerated in the tree which unnecessarily postpone integer solutions (nodes 6 and 8).

From the structure of the network in Fig. 3 it is clear that selecting process 1 implies selecting process 3; likewise selecting **process** 2 also **implies** selecting process 3. Finally, **selecting process 3 implies selecting process 1** and/or process **2. The following logical relationships among the binary** variables can then be stated,

$$\begin{aligned} y_1 &\Rightarrow V_3 \\ y_2 &\Rightarrow V_3 \\ y_3 &\Rightarrow y_1 \vee y_2 \end{aligned} \tag{10}$$

or equivalently as

$$\begin{aligned} &\neg y_1 \vee y_3 \\ &\neg y_2 \vee y_3 \\ &\neg y_3 \vee y_1 \vee y_2 \end{aligned} \tag{11}$$

These relationships can also be expressed in the form of the following inequalities,

$$\begin{aligned} y_3 - y_1 &\geq 0 \\ y_3 - y_2 &\geq 0 \\ y_1 + y_2 - y_3 &\geq 0 \end{aligned} \tag{12}$$

(10), (11) or (12) represent logic cuts for the MILP problem. These cuts can be used either by directly adding the inequalities (12) into the MILP constraint set, or else by performing symbolic inference on (11) when performing a branch and bound search.

The branch and bound tree that results by adding the integer constraints (12) representing the logic cuts for the MILP model is shown in Fig. 5. Note that only 4 nodes must be enumerated, which represents a reduction of 50% with respect **to** the tree in Fig. **4.** Also **note that apart from reducing** the integrality gap, the logic cuts have the effect of eliminating fractional solutions in the tree (nodes 2 and 4 in Fig. 4), and eliminating nodes with zero flow and active processes (nodes 5 and 7 in Fig. 4).

If symbolic inference is performed with the logic cuts in (11), the branch and bound tree shown in Fig. 6 is obtained. Note that this tree also involves the enumeration of only 4 nodes despite the fact that the integrality gap in the LP relaxation is not improved. In this case the logic relations in (11) are used as follows. Since at the root node in one branch we fix jfa = 1* [w]« can also fix jfc = 1. Similarly, in node 2, since yi = jfa = 0 in one branch we can fix jfc = 0; in the other branch since we set jft = 0 and yi = 1, we can fix jte = 1. Thus, in this particular example processing the logic symbolically produces an identical tree as when the logic cuts are included as constraints.

Finally, one can add *violated* logic cuts to the MILP model at the root node before beginning branch and bound, during which the remaining logic cuts are used symbolically. The LP relaxation at the root node has solution *y* = (yi,y2*!ta) = (0,0.444,0.2), which violates the second logic cut in (12). Adding this cut and re-solving the LP yields *y* = (0.244,0.2,0.2), which -violates the first cut in (12). Adding this cut yields y = (0.222,0.222,0.222), which satisfies the remaining logic cut. Branch and bound now generates an identical tree as in Fig. 5. We will see that this strategy can be very effective in some larger examples.

6 Logic Cuts for Processing Networks

In this section we identify a complete set of logic cuts for processing networks with fixed costs. Processing networks arise in applications in which a subset of the mass balances are given by fixed proportions. These include for instance industrial chemical complexes and distillation sequences (see Andrecovich and Westerberg [1]). Fig. 7 presents an example of a chemical

processing network. The objective in these problems is to find a subnetwork that minimizes total variable and fixed costs incurred by the processing units. For simplicity, we consider networks without recycles, although their treatment is similar.

For our purposes, a processing network is a directed acyclic network with two kinds of nodes: *structural nodes* and *unit nodes.* A structural node, represented as a circle in Fig. 7, is a normal network node that conserves flow. A unit node, represented as a square, is a processing node that furthermore requires a fixed construction cost if it is to carry positive flow. We will assume that every source node (node with no inputs) and every sink node (node with no outputs) is a dummy unit node. Unit node i is a *predecessor* of unit *j* if there is a directed path from i to j. Unit node *j* is a *successor of* unit i if i is a predecessor of j. Unit fc is an *immediate successor* of output *j* of unit i (or an *immediate predecessor* of input *j* of *i)* if there is a directed path from output *j* of i to *k* (or from *k* to input *j* of i) containing no other unit.

For each unit node i, there is an associated flow variable Z_i. The flow into each input *j* must be $a_{ij}Z_i$, and flow from each output *j* must be $P_{ij}Z_i$, where $a_{ij} > 0$. We associate each unit node i with a binary variable y_i such that $y_i = 1$ if unit *i* is constructed, and 0 otherwise. d_i (≥ 0) is the fixed cost of constructing unit i. This problem is formulated in the following mixed integer programming model.

$$\begin{aligned} \max \quad & cq - dy \qquad (13)\\ \text{s.t.} \quad & \sum_j q_{ij} = \sum_j q_{ji}, \ \forall i \\ & q_{ji} = \alpha_{ij} Z_i, \ \forall (j,i) \\ & q_{ij} = P_{ij} Z_i, \ \forall (i,j) \\ & Z_i \leq M_i y_i, \ \forall i \\ & Z_i \geq 0, \quad y_i \in \{0,1\}, \quad \forall i \end{aligned}$$

This model has the form of (4) if we let $x = (q, Z)$.

There is no point in constructing a unit that carries no flow. We may therefore ignore solutions with $y_i = 1$ and $Z_i = 0$, even though they may be feasible. As explained earlier, we cannot add mixed-integer constraints that rule out all such solutions, but logic cuts can exclude some of them.

To find all logic cuts for a processing network, we begin with the following lemma.

Lemma 3 *The logical rule*

$$y^* = *(yuV \ldots Vy_{fm}) \tag{14}$$

is a logic ***cut for*** *(IS)* ***if and*** *only* ***if*** $y_{i1} = \cdots = y_{im} = 0$ ***implies*** $Z_i = 0$.

Proof. **Let 5 in Lemma 2 of Section 3 be the set of points *y* satisfying** (14),andletx = $(_9, Z)$.

First assume $y_{i1} = \cdots = y_{im} = 0$ implies $Z_i = 0$. **Take any *tf* £ *S*, and** let **y be the same as *x/* except that $y_r = 0$. Then y € *S* and y ≤ y'. Also for** any (x,y*) € -D, clearly (x,y) € D_y because $Z_r = 0$ implies **that y_r can be** zero. So, (14) is a logic cut by Lemma 2.

Conversely, suppose (14) is a logic cut. Take an (x,^) € *D* **for** which $y'_{i1} = \cdots = y_{im} = 0$. Then either $y_r = 0$, in which case we must have $Z_r = 0$, or $y_r = 1$, in which case y' ^ 5. But in the latter case, there is a **y** € *S* with **y** ≤ *y'* and (x,y) € *D*. But **y** ≤ *y'* and **y** € *S* imply $Z_r = 0$. a

Obviously, there is no flow through unit ***i*** *(Zi* = 0) if we close **all** the immediate predecessors of some input to t, or if we close all immediate successors of some output from t, because each $otij_y/3ij > 0$. **Let us say that (14)** is **a** *premiere cut* if t'i,…, i_m are all the immediate **predecessors of some** input to t, or they are all the immediate successors **of some output from *t*.** We immediately have the following Lemma.

Lemma 4 *Any premiere cut is a logic cut.*

For example *yu* =» (yio V yn) is a premiere cut for Fig. 6, and it is also a logic cut. Not all logic cuts are premiere, as for example the logic cut **!/i7 => *(V2* V y_4).**

We call a logic cut of the form (14) *minimal* if it is not a logic cut when any variable in the consequent is removed. Obviously, a premiere cut is a minimal cut. We have

Lemma 5 *// (14) is a minimal logic cut, then either all units* t'i,. •., t_m *are successors of unit* i, *or they are all predecessors of unit* i.

Proof. Suppose otherwise. Without loss of generality, let $i_1, \ldots, i_r$ be predecessors of t and $t_{r+1}, \ldots, i_m$ successors of $*$, with $r < m$. Since $y_{i_1} \Rightarrow (y_{i_1} \vee \ldots \vee y_{i_r})$ is not a logic cut, setting $y_{i_1} = \ldots = y_{i_r} = 0$ does not (by Lemma 3) block flow into t. Since $y_{i_{r+1}} \Rightarrow (y_{i_{r+1}} \vee \ldots \vee y_{i_m})$ is not a logic cut, setting $y_{i_{r+1}} = \ldots = y_{i_m} = 0$ does not block flow out of t. Thus by Lemma 3 (14) is not a logic cut, contrary to the hypothesis. Q

Lemma 6 *Any logic cut of the form (14) is implied by a finite conjunction of premiere logic cuts.*

Proof. Any logic cut of form (14) is implied by a minimal cut obtained by removing zero or more consequents. We suppose therefore that (14) is a minimal cut. By Lemma 5, we can assume that all the units appearing in the consequent side are predecessors of unit i. (The argument is similar if they are all successors.) The following procedure generates a set of premiere cuts that implies (14).

Let $j_1, \ldots jjk$ be the immediate predecessors of any input to unit t, and let the set F consist of (14). Then (14) is clearly implied by

$$Vi \Rightarrow (y_{i_0} \vee \ldots \vee y_{i_j}), \tag{15}$$

$$Vh \Rightarrow (y_{i_0} \vee \ldots \vee y_{i_m}),\ \forall * \in \{i_1, \ldots, j^*\} \setminus \{t_1, \ldots, t_m\}, \tag{16}$$

where (15) is a premiere cut by Lemma (4). Add the rules (16) to F and delete (14) from F.

For any rule in F, repeat the above. The procedure terminates when F is empty. Since the network has a finite number of units, the procedure will generate a finite set of premiere cuts that implies (14). D

As an example, consider Fig. 7 again, $yn \Rightarrow (y_2 \vee y_4)$ is a logic cut, but not premiere. It is implied by premiere cuts $yn \Rightarrow (y_{i_0} \vee yn)$, $y_{10} \Rightarrow y2$ and

Finally, we show that premiere cuts cut off all integer points that can be cut off by logic cuts of any form.

Theorem 1 *If all premiere cuts are added to (13), then there is no further nonvalid logic cut*

Proof. By Corollary 1, it suffices to show that no feasible x^f is dominated by a feasible $y \wedge y'$. Consider any feasible y'. We wish to show that there is some $(x, y') \in D$ for which no other point $(x, y) \in D$ satisfies $y \leq y'$. To do this we suppose without loss of generality that $y' = (0, e)$, where 0 is a vector of $k (< p)$ zeros and c a vector of $p - k$ ones.

Claim. There are points $(x^{*+\wedge}, I^{\wedge+1}), \ldots_M (x^*, 0, 1P) \in D$, with $x^* = (x^1 Z^f)$ and $f = (y_{k+1}, \ldots, y_p)$, such that each $Z^i > 0$.

Proof of claim. Suppose the claim is false for some $i \in \{k + 1, \ldots_M p\}$. Then $y_i = \ldots = y^* = 0$ implies that $Z_i = 0$, which means by Lemma 3 that $V_i \Rightarrow (y_i \vee \ldots \vee y_j)$ is a logic cut. It is therefore implied by premiere cuts, which contradicts the assumption that y^7 (which has $y_j = 1$ and $y_j = \ldots = y_p = 0$) satisfies all premiere cuts.

Since $y^1 = (0, e)$ satisfies all premiere cuts, the claim implies that $(x^{**1}, 0, c), \ldots, (x^p, 0, c) \in D$. Therefore we can let $(x, 0, e)$ be their convex combination using equal weights, with $x = (x^7, Z)$, and note that $(x, 0, e) \in D$. Since $Z_i^* > 0$ for $i = k + 1, \ldots, p$, there is no point $(x, y) \in D$ with $y \leq y' = (0, c)$ and $y \wedge y^1$. The theorem follows. D

7 Preliminary Numerical Results

In order to test the potential effectiveness of logic cuts, computational results have been obtained for 5 processing network problems. Specifically, we have compared for the first three problems solution requirements in terms of the number of nodes, simplex iterations and CPU time for the following cases:

(a) Solution by Branch and Bound of the MILP model.

(b) Solution by Branch and Bound by addition of all logic cuts in the form of linear inequalities to the MILP.

(c) Solution by performing symbolic inference on the logic cuts within branch and bound enumeration. This was performed by converting the logic cuts to their Disjunctive Normal Form representation, and using as a branching rule the selection of the variable that potentially falsifies the largest number of clauses in order to fix additional 0-1 variables whenever possible [10].

(d) As in (c), but with the addition of violated logic cuts to the MILP at the root node, as described in Section 5.

For the other two problems we compared cases (a) and (c), in the former case with and without a preprocessor.

The numerical tests were performed on an IBM POWER 530 with the OSL optimization model. The symbolic inference method was implemented in OSL with a FORTRAN program through the exit subroutines EKKBRNU and EKKEVNU, for the branching rule and for the fixing of 0-1 variables, respectively. For the case of the preprocessor the routine EKKMPRE was used with the option *nval* = 1. The option performs preprocessing at each node of the tree creating a new matrix with extra rows and tighter bounds for the variables.

The first set of three problems involve the solution of an MILP model for the synthesis of minimum cost distillation sequences for the separation of multicomponent mixture (Andrecovich and Westerberg, [1]). The three problems considered involve the separation of 4, 5 and 6 components. The network for the 4 component system is shown in Fig. 8. The logic cuts that relate the existence of units in the form of prepositional logic are shown in Table 2.

The results for the first three problems, without using OSL's presolve or preprocessing options, are shown in Table 3. Note that in the second column, in which all the logic constraints are included in the model [case(b)], the relaxation gap is significantly reduced, but at the expense of almost doubling the number of rows. Interestingly, exactly the same effect is achieved in the fourth column by adding a much smaller number of violated cuts at the root node [case(d)]. While the branch and bound method with symbolic inference does not require additional inequalities, it requires the largest numer of nodes when compared to cases (b) and (d). Yet it requires significantly fewer nodes than solving the problem without logic cuts. Adding violated cuts to the MILP at the root node reduces the number of nodes still further. For completeness, CPU times have been reported, although they are not very significant given the small size of the problems and the preliminary nature of the implementation scheme in OSL.

Table 4 presents results on two larger problems which are much more difficult to solve. These correspond to heat integrated distillation networks as reported in [10]. Problem 2a only has non-zero fixed charges for the distillation columns, while problem 2b has fixed charges for both columns and heat exchangers. Note that problem 2a could not be solved with OSL after 5000 sec without the preprocessor. With the preprocessor, the solution was ob-

tained in 57.9 sec. In contrast by using the branch and bound with symbolic inference only 14.4 sec were required, due to the reduction in the number of nodes (from 153 to 103) and the number of simplex iterations (7900 to 750). Interestingly in 2b, the solution with the preprocessor required more CPU time than the direct solution of the MILP. However, in terms of number of nodes, simplex iterations and CPU-time the symbolic branch and bound is again the most effective. Although the results of this paper are preliminary, they do suggest that processing the logic cuts symbolically within a branch and cut method may be a promising approach to the optimization of process networks with fixed charges.

8 Future Research

It remains to investigate whether logic cuts are computationally useful in other problem classes, and to describe a complete set of logic cuts for these classes as we have done for processing networks with fixed costs.

One possible application of logic cuts, also network-related, was brought to our attention [4]. The problem is to build tree-shaped utility pipeline network to serve a growing residential or business area. New links incur fixed charges but earn revenue from customers served. Unserved customers provide their own utilities. The solution technique in [4] exploits the fact that building disconnected segments of the tree is feasible but dominated by other solutions. The dominated solutions can be ruled out by logic cuts.

An interesting theoretical problem is to find an algorithm that can, in principle, generate inequalities representing every lope cut (up to equivalence) for an arbitrary MILP. A first step toward achieving this goal might be to find an algorithm that generates all *valid* logic cuts. In a classic paper [3] V. Chvátal solved this problem for traditional valid cuts in pure integer programming problems. J. Hooker [6] solved the problem for valid logic cuts in pure integer programming problems. It is unsolved for MILP problems.

References

[1] M.J. Andrecovich and A.W. Westerberg. An MILP formulation for heat-integrated distillation sequences. *AICHE Journal,* 31:1461-1474, 1985.

[2] E. Balas, S. Ceria, and G. Cornuejols. A cutting plane algorithm for mixed 0-1 programs. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213, USA, 1991.

[3] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305-337,1973.

[4] H. Gröflin, Th.M. Liebling, and A. Prodon. Optimal subtrees and extensions. *Annals of Discrete Mathematics*, 16: 121-127,1982.

[5] F. Harche, J.N. Hooker, and G. Thompson. A computational study of satisfiability algorithms for propositional logic. Technical Report 1991-27, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213 USA, 1991.

[6] J. N. Hooker. Generalized resolution for 0-1 linear inequalities, to appear.

[7] R.G. Jeroslow. Representatibility in mixed integer programming: Chraracterization results. *Discrete Applied Mathematics*, 17:223-243, 1987.

[8] R.G. Jeroslow. *Logic Based Decision Support-Mixed Integer Model Formulation.* Annals of Discrete Mathematics. North-Holland, 1989.

[9] R.G. Jeroslow and J. Wang. Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, 1:167-187, 1990.

[10] R. Raman and I.E. Grossmann. Symbolic Inferences of logic in MILP branch and bound mothed for process synthesis. AICHE Annual Meeting, Los Angeles. (1991).

[11] H.D. Sherali and W.P. Adams. Hierarchy of relaxations and convex hull characterizations for mixed-integer zero one programming problems. Technical report, University of Virginia, 1989.

[12] T.J. Van Roy and L.A. Wolsey. Solving mixed integer programs by automatic reformulation. *Operations Research*, 35:45-57, 1987.

[13] H.P. Williams. *Model Building in Mathematical Programming.* Wiley, Chichester, 1988.

Table 1. Logic cuts for the processing network in Fig. 8

$V2 + y_3 + y4 - yi$	$\geq$	0
$yi - y2$	$\geq$	0
$yi - y3$	$\geq$	0
$yi - y4$	$\geq$	0
$ys + ye - y7$	$\geq$	0
$y7 - ys$	$\geq$	0
$y? - ye$	$\geq$	0
$ys + y8 + yio - yi$	$\geq$	0
$ys + y9 + yio - y7$	$\geq$	0

Table 2. Logic cuts for a four component separation network

yi	$\Leftrightarrow$	$y3 \vee VA$	ye	$\Rightarrow$	yg
y2	$\Leftrightarrow$	$ye \vee yi$	y7	$\Rightarrow$	yio
y3	$\Rightarrow$	ys	y8	$\Rightarrow$	$y_3 \vee ys$
y4	$\Rightarrow$	y9	y9	$\Rightarrow$	$y_4 \vee ye$
ys	$\Rightarrow$	$ys \wedge yio$	yio	$\Rightarrow$	$ys \vee y7$

Table 3. Separation Problems

Number of Components	Original Formulation	Logic Cute Added to MILP	Logic Cuts Used Symbolically	Violated Logic Cuts Added to MILP at Root Node[1]
FOUR—31 vaLriables (10 biinary)				
Rows	**27**	**49**	**27**	**35 (8 cuts)**
LP optimum	**3601.6**	**3625.8**	**3601.6**	**3625.8**
MILP optimum	**3625.8**	**3625.8**	**3625.8**	**3625.8**
Nodes	**12**	**0**	**8**	**0**
Iterations	**26**	**21**	**20**	**16**
CPU **time***	**0.54**	**0.32**	**0.75**	**0.29**
FIVE—61 variables (20 **binary)**				
Rows	**51**	**90**	**51**	**57 (6 cuts)**
LP optimum	**4262.7**	**4278.1**	**4262.7**	**4278.1**
MILP optimum	**4304.1**	**4304.1**	**4304.1**	**4304.1**
Nodes	**52**	**2**	**8**	**4**
Iterations	**179**	**48**	**35**	**76**
CPU time*	**1.23**	**0.49**	**0.92**	**0.65**
SIX—106 vanskbles **(35 binary)**				
Rows	**86**	**156**	**86**	**97 (11 cuts)**
LP optimum	**18114.7**	**18161.6**	**18114.7**	**18161.6**
MILP optimum	**18170.3**	**18170.3**	**18170.3**	**18170.3**
Nodes	**141**	**8**	**14**	**4**
Iterations	**386**	**219**	**50**	**40**
CPU time*	**3.46**	**1.18**	**1.44**	**0.87**

• sec on IBM POWER 530 with OSL.

f Remaining logic cuts are used symbolically.

Table 4. Heat Integrated Distillation Network Problems

147 constraints 261 variables (100 binary)	Original Formulation		Logic Cuts Used Symbolically
	without Preprocessor	with Preprocessor	
Problem 2a			
LP Optimum	388.3	388.3	388.3
MILP Optimum	23627.5	23627.5	23627.5
Nodes	$>10^5$	153	103
Iterations	$> 10^6$	7900	750
CPU time'	> 5,000	57.95	14.4
Problem 2b			
LP Optimum	390.4	390.4	388.7
MILP Optimum	3655.8	3655.8	3655.8
Nodes	448	235	101
Iterations	1651	28883	792
CPU time*	22.06	169.39	14.63

*** sec on IBM POWER 530 with OSL.**

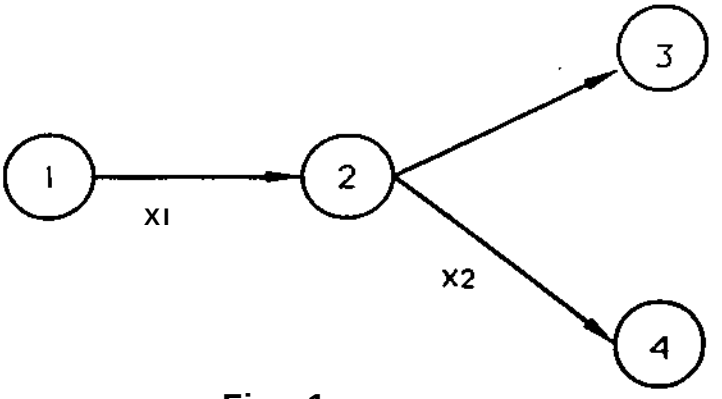


Fig. 1

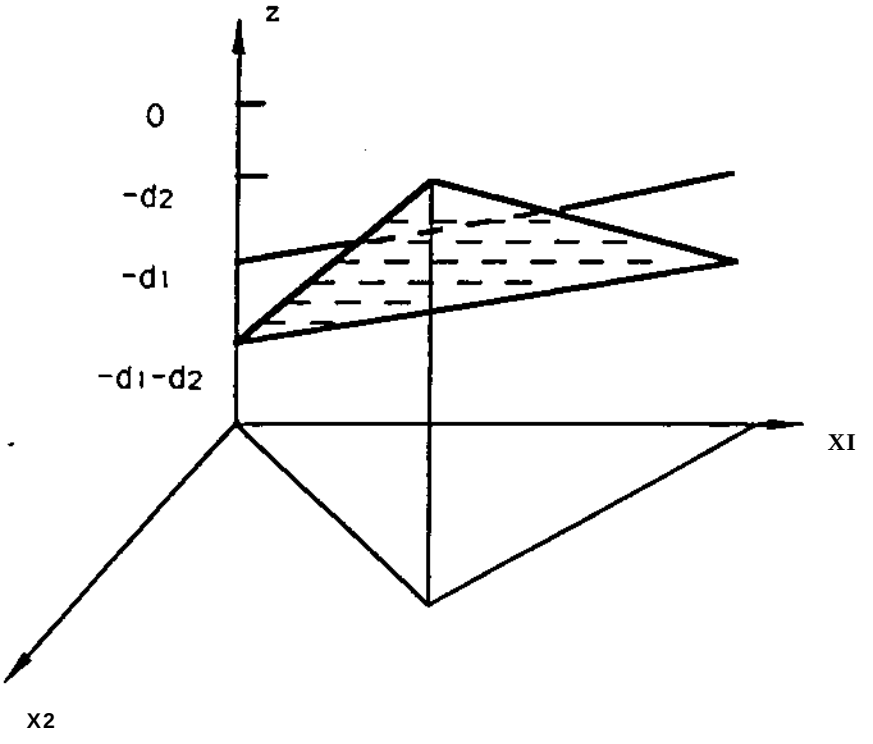


Fig. 2

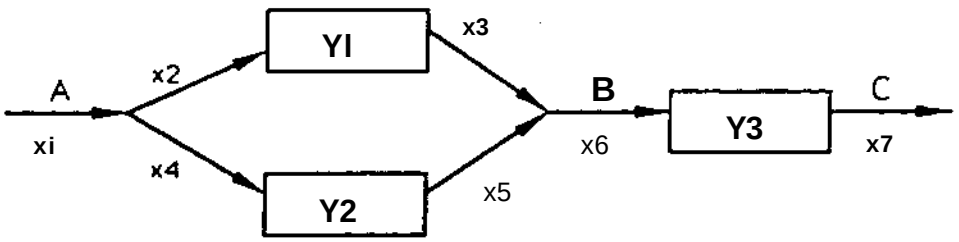


Fig. 3 Illustration of a processing netuiork

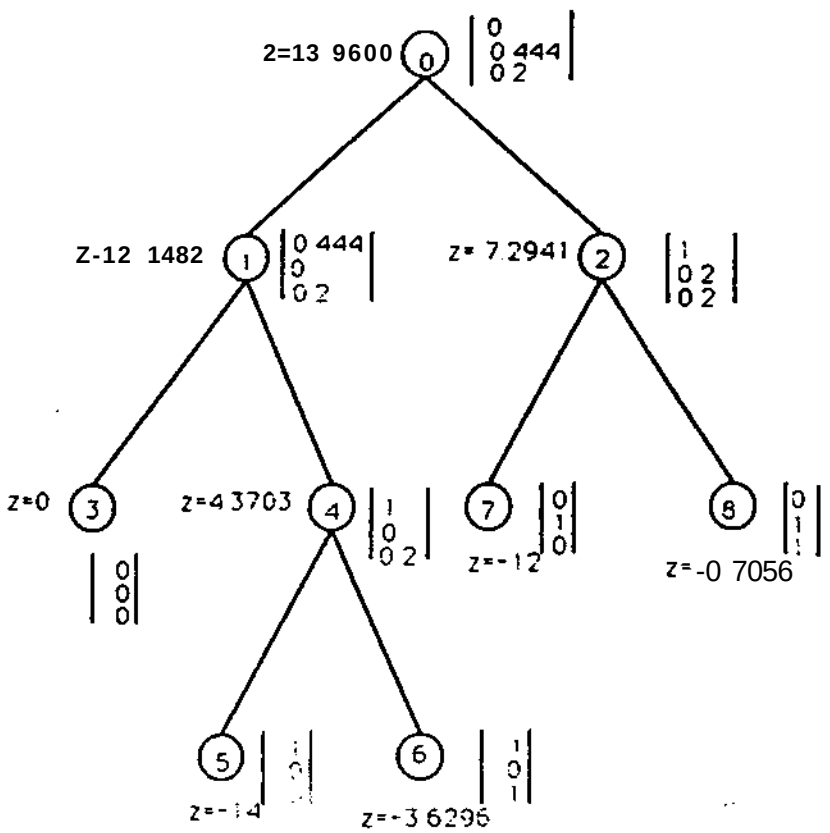


Fig. 4 Search tree for original model

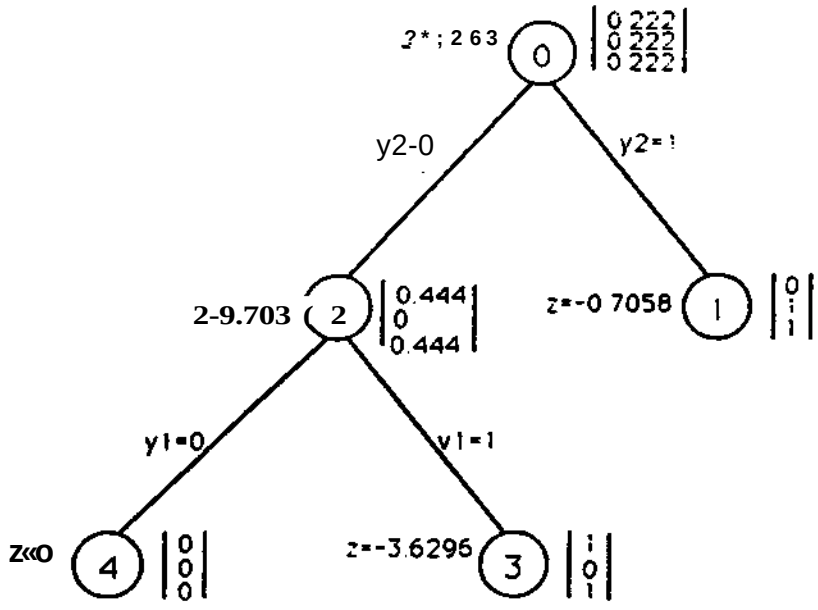
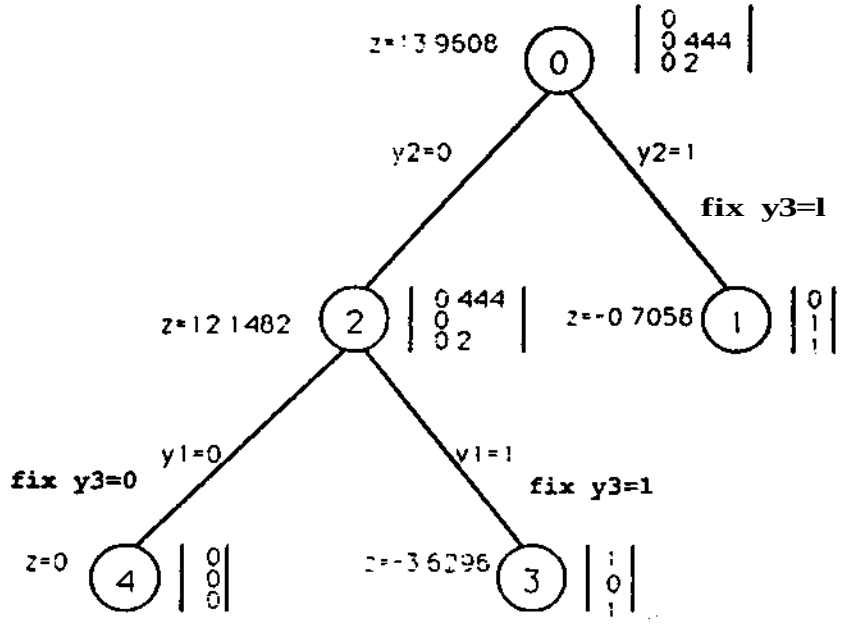


Fig. 5 Search tree when logic cuts ore added to model



Fig, 6 Search tree uuith symbolic inference in the model

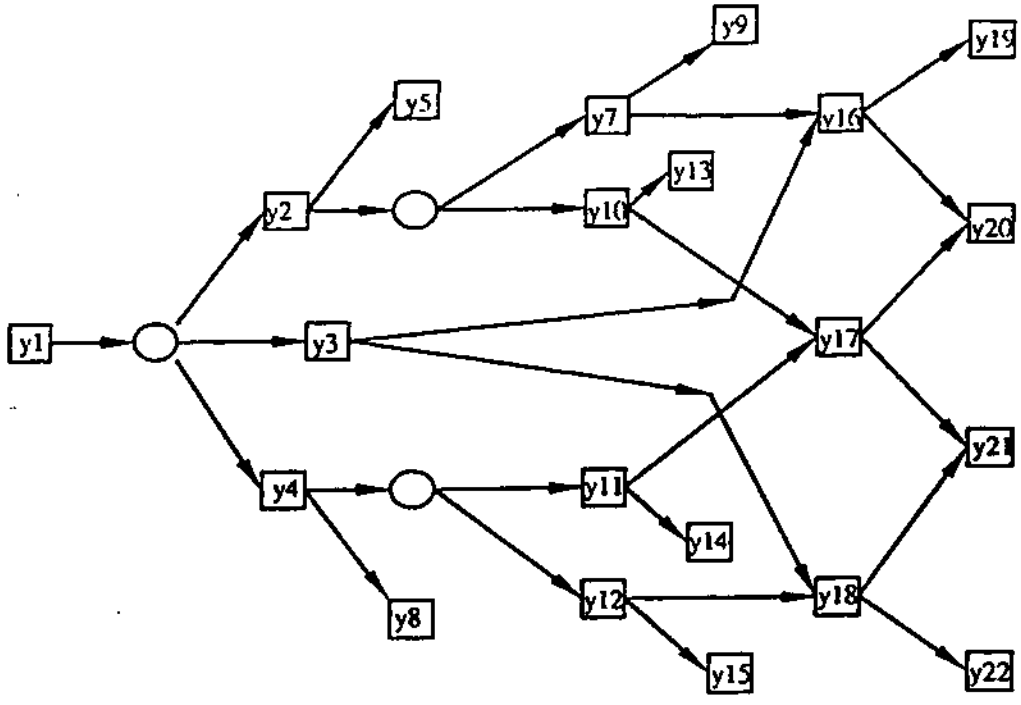


Fig. 7 Ewample of a processing network

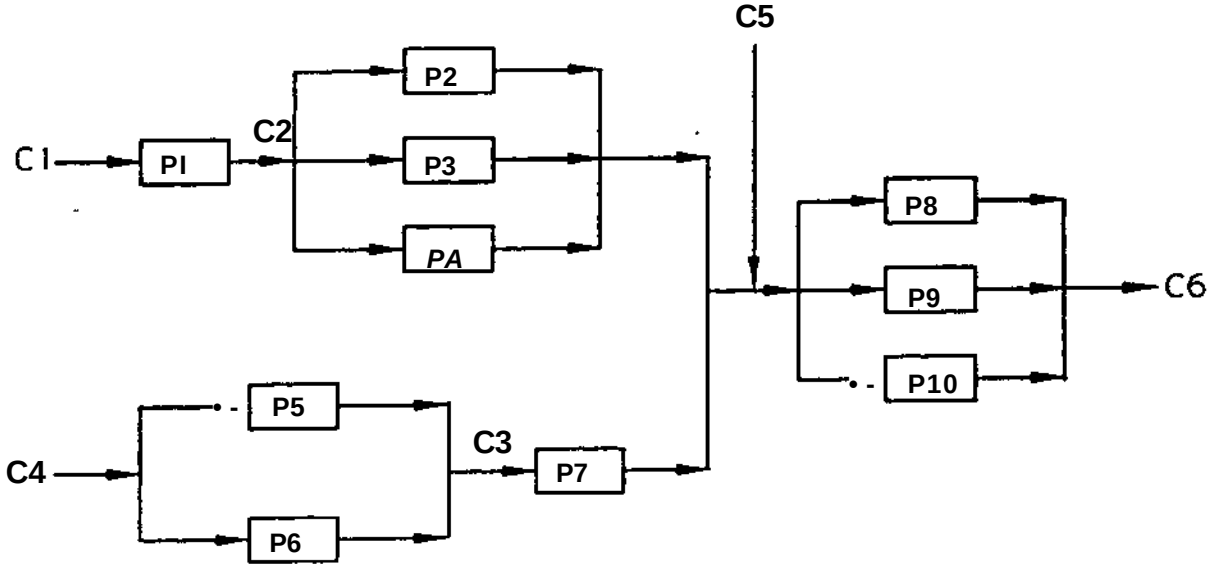


Fig. 8 Chemical processing network

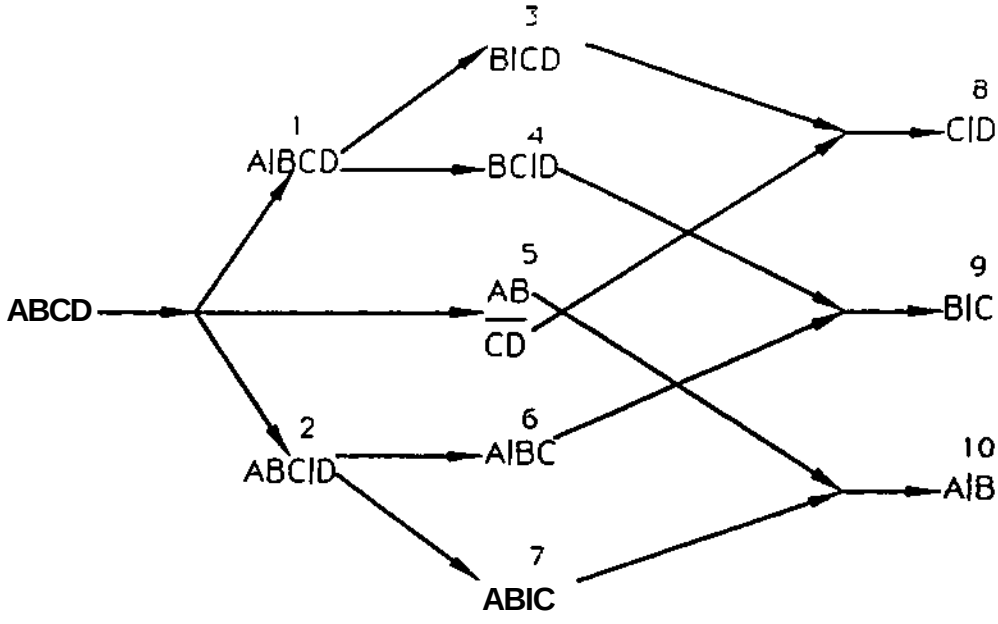


Fig.9 Network for four component separation