European Journal of Operational Research 48 (1990) 362–371
North-Holland

Theory and Methodology

An algorithm for disjunctive programs

Nicholas Beaumont *
Monash University, Clayton, Victoria 3168, Australia

Abstract: A disjunctive program is a linear program complicated by disjunctions, that is, sets of constraints of which at least one must be true. The commonest example is the fixed cost problem in which *either* production of an item is zero *or* a fixed cost is incurred. The conventional solution technique is to re-express each disjunction in terms of binary variables and solve the resulting mixed integer program. This paper demonstrates that this re-expression is unnecessary: it is possible and usually advantageous to arbitrate on the logical relationships amongst constraints themselves. The advantages include easier matrix generation, smaller nodal problems and the availability of good arbitration criteria. An example is given and computational results are summarized.

Keywords: Integer programming, disjunctive programming

1. Introduction

A logical program entails optimization subject to logical combinations of constraints, [14] contains a number of problems which can be neatly and naturally expressed as logical programs. The commonest example is one in which either a setup cost is incurred OR production is zero. The conventional method of solving such problems is to introduce binary variables (hereafter called logical variables) and to re-express logical relationships amongst constraints in terms of ordinary constraints and logical variables [15]. The resulting mixed integer program is usually solved by branch and bound algorithm, that is, by relaxing the problem, traversing a tree formed by arbitrating the logical variables and solving the subproblems thus formed.

This method has a number of disadvantages:

- Matrix generation is complicated by the need to re-express logical relationships in terms of logical variables and ordinary constraints.
- It enlarges the problem by introducing extra rows, columns and coefficients.
- It is wasteful to express, for example, a binary disjunction of constraints as two rows knowing a priori that in any integer feasible solution at least one of the rows will be slack.

These considerations lead to a re-examination of the idea [11] of a branch and bound algorithm which arbitrates logical constraints (a logical constraint is a logical combination of 'simple' constraints) instead of logical variables. For brevity, only disjunctions will be discussed, that is sets of constraints of which at least one member must be true.

* Now at Department MCOR, Footscray Institute of Technology, Vic 3011, Australia.

Received September 1988; revised May 1989

A disjunctive program has the form

$$\begin{aligned}
\text{(DP)}\quad &\text{maximize} && \sum_{j\in N} c_j x_j \\
&\text{subject to} && \sum_{j\in N} a_{ij}x_j \leqslant b_i, \quad i \in M, \\
& && 0 \leqslant x_j \leqslant m_j, \quad j \in N, \\
& && \bigvee_{i\in D(k)} \left[\sum_{j\in N} a_{ij}x_j \leqslant b_i \right], \quad k \in K, \qquad (1)
\end{aligned}$$

where M, N, $D(k)$, $k \in K$, and K are index sets. Formula (1) defines a disjunction.

2. Surrogates

A mixed integer program (MIP) is relaxed by replacing the condition

$$\delta = 0 \vee \delta = 1 \quad \text{by} \quad 0 \leqslant \delta \leqslant 1, \tag{2}$$

which is the best possible relaxation. A surrogate of a disjunction is defined as a simple constraint which is implied by the variables' bounds and any term of the disjunction. A disjunctive program (DP) can then be relaxed by replacing each disjunction by one of its surrogates. The best possible surrogate is a facet of the convex hull of the feasible region of the disjunction, but such facets are computationally difficult to find [1, p. 21, 2,13].

The following methods of obtaining a surrogate for

$$\bigvee_{i\in D} \sum_{j\in N} a_{ij}x_j \leqslant b_i \tag{3}$$

(where D is an index set with ordinality d) and $0 \leqslant x_j \leqslant m_j$ are available as options in the computer program which implements the disjunctive algorithm.

2.1. Elementary

This method is described because it is analogous to, and as tight as the relaxation (2). The MIP formulation of (3) is

$$\sum_{j\in N} a_{ij}x_j \leqslant b_i + u_i(1-\delta_i), \quad i \in D, \tag{4}$$

$$\sum_{i\in D} \delta_i \geqslant 1, \tag{5}$$

where δ_i, $i \in D$, is a logical variable and

$$u_i = \sum_{j\in N} m_j a_{ij}^+ - b_i, \quad i \in D, \tag{6}$$

where $a^+ = \max(0, a)$. If a u_i is not positive then the constraint i is redundant and the disjunction is satisfied. Elimination of the δ_i, $i \in D$, yields

$$\sum_{i\in D} 1/u_i \sum_{j\in N} a_{ij}x_j \leqslant d - 1 + \sum_{i\in D} b_i/u_i, \tag{7}$$

(where $d = |D|$) which is the required surrogate.

2.1.1. Feasible regions

It is now shown that the disjunctive relaxation (7) is as strong as the relaxation of the conventional formulation. Let FA (spanned by the vectors $\boldsymbol{x} \equiv (x_j,\ j \in N)$ and $\boldsymbol{\delta} \equiv (\delta_i,\ i \in D)$ denote the feasible region (defined by (4), (5) and $0 \leqslant \delta \leqslant 1$) of the conventional relaxation. Let FD (spanned by $\boldsymbol{x}$) denote the feasible region (defined by (7)) of the disjunctive relaxation. Let FC denote the projection of FA onto the x plane. It is now shown that FC = FD.

By construction, FC $\subseteq$ FD. It is now shown that for any point $x \in$ FD a $\boldsymbol{\delta}$ can be constructed so that $(\boldsymbol{x}, \boldsymbol{\delta}) \in$ FA. Let

$$f_i = \left(b_i + u_i - \sum_{j\in N} a_{ij}x_j \right) / u_i, \quad i \in D, \tag{8}$$

if $\delta_i \leqslant f_i$ it is clear that (4) is satisfied. By eliminating

$$\sum_{i\in D} \frac{1}{u_i} \sum_{j\in N} a_{ij}x_j \tag{9}$$

from (4) and (8), $\Sigma_{i\in D} f_i \geqslant 1$ is obtained. Combining (6) and (8) yields

$$u_i f_i = \sum_{j\in N} a_{ij}^+ m_j - \sum_{j\in N} a_{ij}x_j, \quad i \in D, \tag{10}$$

whence $f_i \geqslant 0$, $i \in D$. Let

$$f_i' = \frac{f_i}{\Sigma_{i\in D} f_i}, \quad i \in D. \tag{11}$$

Any value of δ_i such that $f_i' \leqslant \delta_i \leqslant f_i$, $i \in D$, satisfied (4), (5) and $0 \leqslant \delta_i \leqslant 1$, $i \in D$. Therefore FD $\subseteq$ FC, FD = FC, and the two relaxations have equal strength.

2.1.2. Shadow prices

It is now shown that the shadow prices of (5) and (7) are the same. Consider the system

$$\text{(P)}\quad \text{maximize} \quad \sum_{j\in N} c_j x_j$$

subject to

$$\sum_{j\in N} a_{ij}x_j \leqslant b_i, \quad i\in M,$$

$$x_j \leqslant m_j, \quad j\in N,$$

$$\bigvee_{l\in L} \sum_{j\in N} a_{ij}x_j \leqslant b_i, \quad i\in D_l, \quad l\in L,$$

where L is an index set. The conventional relaxation is

$$\text{(CR)}\quad \text{maximize} \quad \sum_{j\in N} c_j x_j$$

subject to

$$\sum_{j\in N} a_{ij}x_j \leqslant b_i, \quad i\in M, \tag{12}$$

$$x_j \leqslant m_j, \quad j\in N, \tag{13}$$

$$\sum_{j\in N} a_{ij}x_j + u_i\delta_{il} \leqslant b_i + u_i, \quad i\in D_l, \quad l\in L, \tag{14}$$

$$-\sum_{i\in D_l} \delta_{il} \leqslant -1, \quad l\in L, \tag{15}$$

$$\delta_{il} \leqslant 1, \quad i\in D_l, \quad l\in L, \tag{16}$$

where M, N, L and D_l are index sets. M and D_l must be disjoint for each $l\in L$, D_l has ordinality d_l, and all variables are non-negative.

Suppose nonnegative variables y_i, z_j, p_{il}, q_l and q_{il} for $i\in M$, $j\in N$ and $l\in L$ are associated with the rows (12), (13), (14), (15) and (16) respectively. Because of (11), (16) is redundant, and all $q_{il}=0$. The dual of (CR), denoted by (DCR) is

$$\text{(DCR)}\quad \text{minimize} \quad \sum_{i\in M} b_i y_i + \sum_{j\in N} m_j z_j + \sum_{l\in L}\sum_{i\in D_l} (b_i+u_i)p_{il} - \sum_{l\in L} q_l \tag{17}$$

subject to

$$\sum_{i\in M} a_{ij}y_i + z_j + \sum_{l\in L}\sum_{i\in D} a_{ij}p_{il} \geqslant c_j, \quad j\in N,$$

$$u_i p_{il} - q_l \geqslant 0, \quad i\in D_l, \quad l\in L. \tag{18}$$

The disjunctive relaxation of (P) is

$$\text{(DR)}\quad \text{maximize} \quad \sum_{j\in N} c_j x_j$$

subject to

$$\sum_{j\in N} a_{ij}x_j \leqslant b_i, \quad i\in M, \tag{19}$$

$$x_j \leqslant m_j, \quad j\in N, \tag{20}$$

$$\sum_{i\in D_l} \frac{1}{u_i} \sum_{j\in N} a_{ij}x_j \leqslant d_l - 1 + \sum_{i\in D_l} \frac{b_i}{u_i}, \quad l\in L. \tag{21}$$

If the dual variables for the constraints (19), (20) and (21) are Y_i, $i\in M$, Z_j, $j\in N$, and Q_l, $l\in D_l$, respectively, the dual of (DR) is

$$\text{(DDR)}\quad \text{minimize} \quad \sum_{i\in M} b_i Y_i + \sum_{j\in N} m_j Z_j + \sum_{l\in L}\Big(d_l - 1 + \sum_{i\in D_l} b_i/u_i\Big)Q_l \tag{22}$$

subject to

$$\sum_{i\in M} a_{ij}Y_i + Z_j + \sum_{i\in D_l} (Q_l/u_i)a_{ij} \geqslant c_j, \quad j\in N, \quad l\in L. \tag{23}$$

By considering the direction of optimization in (DCR) it can be seen that (18) must be an equality. Drop this equation and replace p_{il} wherever it appears in (DCR) by q_l/u_i (this is always possible as u_i is, by construction, positive). The third term of (17) can therefore be written as

$$\sum_{l\in L}\sum_{i\in D_l}\left(\frac{b_i q_l}{u_i} + q_l\right) \quad \text{or} \quad \sum_{l\in L} q_l \sum_{i\in D_l}\left(\frac{b_i}{u_i} + 1\right) \tag{24}$$

or

$$\sum_{l\in L}\left(d_l q_l + q_l \sum_{i\in D_l}\left(\frac{b_i}{u_i}\right)\right). \tag{25}$$

By comparing (22) and (25) and noting that the objective function value of (DDR) is equal to that of (DCR), it can be seen that $q_l = Q_l$, $l\in L$; i.e. that the shadow prices of (15) and (21) are the same. This is commented on in Section 4.

2.1.3. *Efficiency*

There is no guarantee that (5) or its equivalent (7) will be 'efficient' in the sense of being a good approximation to a facet of the convex hull of the feasible region. That this method sometimes gives very poor surrogates is exemplified by an example from [7, p. 183]:

$$(x_1 - x_2 \leqslant -100) \vee (-x_1 + x_2 \leqslant -100),$$
$$0 \leqslant x_1,\ x_2 \leqslant 200, \tag{26}$$

whose surrogate (7) is $0x_1 + 0x_2 \leqslant 100$. Nevertheless, in some simple but common cases such as $x_1 = 0 \vee x_2 = 0$, the elementary method gives a facetal surrogate.

Any disjunction of form

$$\sum_{j\in N} a_j x_j \leqslant b_1 \vee \sum_{j\in N} a_j x_j \geqslant b_2, \tag{27}$$

where $b_1 < b_2$ is said to comprise 'antiparallel' terms. Such forms arise, for example, in some job scheduling problems. It is now shown that neither (5) nor (7) can be binding for such disjunctions.

Assume that upper bound substitutions have been applied where necessary so that $a_j \geqslant 0$, $j \in N$. Define $M \equiv \Sigma_{j\in N} a_j m_j$, and avoid trivialities by assuming $M \geqslant b_2 > b_1 \geqslant 0$. Applying (7) to (27) yields

$$\sum_{j\in N} a_j x_j (b_2 - M + b_1) \leqslant b_1 b_2. \tag{28}$$

This surrogate is parallel to the terms of the disjunction and it is easily shown that the points $(0,\ldots,0)$ and $(m_1,\ldots,m_n)$ and consequently every point of the feasible region satisfies it. As FC = FD so does (5).

2.2. *Generalised binary variables*

The form $x \leqslant c_1 \vee x \geqslant c_2$ where $0 \leqslant c_1 < c_2 \leqslant m$ and $0 \leqslant x \leqslant m$ arises quite frequently, for example in blending problems (by setting $z = \Sigma_{j\in N} a_j x_j$, (27) can be reduced to this form). It is clear that the best possible surrogate is the bound interval $0 \leqslant x \leqslant m$. It efficient to be able to relax and arbitrate this kind of disjunction simply by changing a variable's bounds. Setting the bounds on x to $[0,\ m]$ or (say) $[0,\ c_1]$ respectively relaxes or arbitrates the disjunction. No rows or coefficients are required for its expression in this form, whereas the conventional formulation would entail two rows, a logical variable and up to six coefficients (including RHS values). This form could be used to input binary (conceivably general integer) variables ($x = 0 \vee x = 1$).

The concept of a binary variable can be extended to include variables whose domain comprises two disjoint intervals. It is easy to extend MIP codes appropriately [3].

2.3. *The most general surrogate*

A basis for finding all possible surrogates of (3) is a result in [1, p.8] which is repeated here except that the notation is changed slightly and some inequalities are divided by -1.

Balas's theorem. *The inequality $\alpha x \leqslant \beta$ is a surrogate of*

$$\bigvee_{i\in D} (A^i x \leqslant b^i), \quad x \geqslant 0,$$

if and only if there exist $\psi^i \geqslant 0$, $i \in D^*$, such that

$$\alpha \leqslant \psi^i A^i, \quad i \in D^*, \tag{29}$$

$$\beta \geqslant \psi^i b^i, \quad i \in D^*, \tag{30}$$

where A^i is an $m_i \times n$ matrix, x is an n-vector, α is $1 \times n$, β is a scalar, b^i is an m_i-vector, ψ^i is $1 \times m_i$, and D^* is the subset of D for which the terms $i \in D$ of the disjunction are feasible.

Suppose that all the A^i have $n+1$ rows, that the first n rows of $A^i x \leqslant b^i$, $i \in D$, are common and represent upper bounds m_j on x_j, $j \in N$, and that the last row of each term is $\Sigma_{j\in N} a_{ij} x_j \leqslant b_i$. It is now easy to determine which members of D are infeasible; these can can be dropped from D, and thereafter D can be set equal to D^*.

It follows that by partitioning A^i, ψ^i and b^i into

$$\begin{pmatrix} I_n \\ a^i \end{pmatrix}, \qquad (\theta^i \quad \phi^i) \quad \text{and} \quad \begin{pmatrix} m \\ b_i \end{pmatrix},$$

respectively, (29) and (30) can be rewritten as

$$a \leqslant \theta^i + \phi^i i, \quad i \in D,$$

$$\beta \geqslant \theta^i m + \phi^i b_i, \quad i \in D,$$

where a^i is $1 \times n$, I_n is the identity matrix of order n, θ^i is $1 \times n$, m is an n-vector comprising the variables' upper bounds and ϕ^i and b_i are scalars. By setting the parameters θ^i and ϕ^i, each and every valid surrogate of a disjunction can be obtained.

Having chosen ϕ^i, a simple choice is

$$\alpha_j = \max_{i \in D}\left(a_{ij}\phi^i\right), \quad j \in N, \tag{31}$$

whence

$$\theta^i_j = \alpha_j - a_{ij}\phi^i, \quad j \in N, \quad i \in D,$$

$$\beta = \max_{i \in D}\left(\sum_{j \in N} m_j\theta^i_j + \phi^i b_i\right). \tag{32}$$

This will probably give a surrogate with a high proportion of positive coefficients if the a_{ij} are symmetricly distributed about zero.

Consideration is now given to choosing the values of the ϕ^i. Suppose the α_j have been chosen as in (31) and rewrite (32) as

$$\beta = \sum_{j \in N} m_j\alpha_j + \max_{i \in D}\left[\phi^i\left(b_i - \sum_{j \in N} a_{ij}m_j\right)\right].$$

It seems rational to make β as small as possible by choosing

$$\phi^i = 1/\left(\sum_{j \in N} a_{ij}m_j - b_i\right),$$

provided that the denominator is positive. In the implementation, if the denominator is not positive, ϕ^i is set equal to $1/(\Sigma_{j \in N} a^+_{ij}m_j - b_i)$. This option has been lightly tested and usually gives surrogates at least as tight as (7). Another (less satisfactory) option is to set

$$\phi^i = -1/\left(\sum_{j \in N} a_{ij}m_j - b_i\right)$$

if the denominator is negative.

If $\Sigma_{j \in N} a_{ij}m_j \geqslant b_i$ for all $i \in D$, then, as is now proved, this method gives a facetal surrogate. The surrogate is $\alpha x \leqslant \beta$ where $\alpha_j = \max_{i \in D}(a_{ij}\phi^i)$, $j \in N$, and $\beta = \Sigma_{j \in N} m_j\alpha_j - 1 \geqslant 0$. This surrogate cuts the line $L_k \equiv (x_j = m_j,\ j \in N\backslash k)$, $k \in N$, at $(x_k,\ m_j \in N \setminus k)$ where x_k is given by

$$\alpha x_k + \sum_{j \in N} m_j\alpha_j - m_k\alpha_k = \beta = \sum_{j \in N} m_j\alpha_j - 1$$

or

$$x_k = m_k - 1/\alpha_k.$$

The constraint $\Sigma_{j \in N} a_{ij}x_j = b_i$ cuts L_k at

$$x_k = m_k - \frac{b_i - \Sigma_{j \in N} a_{ij}m_j}{a_{ik}}$$
$$= m_k - 1/(a_{ik}\phi_k).$$

Let

$$x^*_k = \max_{k \in N} x_k = m_k - 1/\max_{i \in D}(a_{ik}\phi_k)$$
$$= m_k - 1/\alpha_k = x_k.$$

When applied to (26), this method yields the facetal surrogate $x_1 + x_2 \leqslant 300$.

2.4. User supplied constraints

The final option is to suppress the automatic production of a surrogate. The user may find it more fruitful to include surrogates appropriate to the particular problem in the input file.

3. The disjunctive algorithm

The following algorithm solves disjunctive programs. It is analogous to branch and bound algorithms used to solve mixed integer programs.

Step 0. Form a relaxation of the original (maximization) problem by replacing each disjunction by a surrogate. U, the set of unresolved problems, initially comprises this problem alone. Set BEST $= -\infty$.

Do Steps 1, 2 and 3 **until** U is empty:

Step 1. Remove a problem (denoted by P) from U and solve (or fathom) it as a linear program denoting its objective function value by z (setting $z = -\infty$ if the problem is infeasible).

Step 2. If $x >$ BEST and this is a disjunctive solution, set BEST $= z$ and store the solution.

Step 3. If $z >$ BEST and this is not a disjunctive solution, choose a disjunction and add to U those problems formed by arbitrating the chosen entity in P.

Step 4. The optimal feasible disjunctive solution is the last solution (if any) stored in Step 2. If no solution has been stored the problem is infeasible.

Table 1 summarizes the complete analogy between the conventional and disjunctive algorithms

Table 1
Analogy between the conventional and disjunctive algorithms

Feature	Conventional algorithm	Disjunctive algorithm
Relaxation	Replace $\delta = 0 \vee \delta = 1$ by $0 \leqslant \delta \leqslant 1$	Replace the disjunction by its surrogate
Arbitration	Set a chosen logical variable to 1	Replace a chosen disjunction by one of its terms
Feasible solution	All logical variables are either 0 or 1	At least one term of each disjunction is satisfied

when applied to disjunctions of simple constraints (δ is logical variable which is arbitrated upwards).

4. Criteria

As in the conventional algorithm the total number of nodes which have to be examined is much affected by the way in which the tree is built and traversed. The questions arising in the disjunctive approach are:

- Which problem to select from the set of unresolved problems.
- Which disjunction to arbitrate.
- Which term of the disjunction is to replace the surrogate.

Three criteria suggested by analogy are penalties obtained by anticipating the first dual iteration, pseudocosts and user supplied priorities (see [6] for details and further references).

Rado [11] suggests a *Maximin* criterion. For each disjunction, note the term which has the smallest 'infeasibility' and let this be (by definition) the infeasibility of the disjunction. Arbitrate on the disjunction which has the largest infeasibility. The infeasibility is defined as $\sum_{j \in N} a_j x_j - b$ for a constraint of form $\sum_{j \in N} a_j x_j \leqslant b$ and analogously for other forms.

A *surrogate* criterion is suggested:

1. Arbitrate on the infeasible disjunction whose surrogate has the highest absolute non-zero shadow price.

2. If no disjunction is chosen in (1), arbitrate on the infeasible disjunction whose surrogate has the smallest slack.

3. In either case select the term with the smallest infeasibility gap.

For the test problems this criterion was effective, especially as it always gave good early solutions. The maximin criterion usually required fewer iterations and more CPU time.

The significance of this fact is that, for some kinds of disjunctive programs, the shadow prices (Q_l) of the constraints (21) are good guides as to which disjunction should next be arbitrated. This reinforces the intuitively attractive idea that, in some MIPs, it may be advantageous to arbitrate a variable with a large coefficient in a constraint which has a high shadow price.

Dimensional considerations suggested using a 'scalar product' of the shadow price and infeasibility, but this worsened performance in the test problems used. It seemed rational to re-express the infeasibility gap of a term of a disjunction in relative terms (for example, where the term was a bound, the relative infeasibility was obtained by dividing the absolute infeasibility by the difference between the variable's upper and lower bounds). This change had little impact on solution times, perhaps because the test problems were fairly well scaled.

Because matrices are invariably stored in column order, the two criteria suggested would be difficult to implement in conventional algorithms.

5. An example

The following example illustrates that, for some problems at least, formulation in disjunctive terms is easier and more compact than the conventional formulation. At the very least, the user does not have to calculate the 'big M's'.

5.1. Description

This problem [5, p. 393] entails scheduling buses which link with five departing and arriving aricraft so as to minimize the total cost. The total cost comprises a fixed cost of \$25 for each bus making the trip, and a variable cost of \$0.15/minute for time spent waiting at the airport. The flight departure and arrival times are denoted by A_i and B_k respectively. Buses have to wait at least 10 minutes at the airport. A flight departure must be serviced by a bus arriving at the airport between 45 and 20 minutes before the scheduled departure time. A flight arrival must be serviced by a bus leaving the airport between 15 and 30 minutes after the scheduled arrival time. The formulation requires an upper bound (M) on all the bus trip times, Daellenbach and George [5] use 2000, an unnecessarily large value retained here to facilitate comparisons, and give sets of viable bus trip and flight combinations, thereby excluding implausible combinations such as a bus delivering passengers to the first flight and waiting at the airport to pick up passengers from the last flight.

5.2. *Conventional formulation*

The variables are:

x_j, y_j : the times at which trip j arrives and leaves the airport,

w_j : the waiting time for trip j,

$$\delta_j = \begin{cases} 1 & \text{if trip } j \text{ is run,} \\ 0 & \text{otherwise,} \end{cases}$$
for all j;

$$\beta_{kj} = \begin{cases} 1 & \text{if trip } j \text{ services arrival } k, \\ 0 & \text{otherwise;} \end{cases}$$
for all j, k;

$$\gamma_{ij} = \begin{cases} 1 & \text{if trip } j \text{ services departure } i, \\ 0 & \text{otherwise,} \end{cases}$$
for all i, j.

The constraints are:

$x_j \leqslant M\delta_j$ for all j

(the departure time is zero unless the bus runs),

$x_j \leqslant M$ for all j

(redundant, but hastens solution),

$w_j \geqslant 10\delta_j$ for all j,

$w_j - y_j + x_j = 0$ for all j

(the definition of the waiting times),

$x_j + (M + 20)\gamma_{ij} \leqslant M + A_i$

for all plausible combinations of i, j,

$-x_j + (M - 45)\gamma_{ij} \leqslant M - A_i$

for all plausible combinations of i, j (a bus must arrive between 45 and 20 minutes before each departing flight),

$\sum_j \gamma_{ij} = 1$ for all i

(the flight must be served by one bus),

$y_j + (M - 30)\beta_{kj} \leqslant M + B_k$

for all plausible combinations of k, j,

$-y_j + (M + 15)\beta_{kj} \leqslant M - B_k$

for all plausible combinations of k, j.
Similarly, each arriving flight must be met by a bus departing between 15 and 30 minutes after its arrival time.

$\sum_j \beta_{kj} = 1$ for all k.

The objective function is

minimize $\sum_j (25\delta_j + 0.15w_j)$.

5.3. *Disjunctive formulation*

One new variable c_j, the fixed cost incurred if trip j is taken, is required. No binary variables are needed. The constraints are:

$x_j \leqslant M$ for all j,

$w_j \geqslant 10$ for all j,

$w_j - y_j + x_j = 0$ for all j,

$x_j \leqslant 0 \vee c_j = 25$ for all j

(the bus is not used or a fixed cost is incurred),

$\bigvee_j (A_i - 45 \leqslant x_j \leqslant A_i - 20)$

for each i and the corresponding plausible values of j,

$\bigvee_j (B_k + 15 \leqslant y_j \leqslant B_k + 30)$

for each k and the corresponding plausible values of j.
These two constraints ensure that each flight departure and arrival is met by at least one bus. The objective function is

minimize $\sum_j (c_j + 0.15w_j)$.

Table 2
Summary of results for five test problems and their approaches

	Transport		Mining		Bus		Blending		Schedule	
	Conv.	Disj.	Conv.	Disj.	Conv.	Disj.	Conv.	Disj.	Conv.	Disj.
Binaries or disjunctions	64	64	100	60	29	17	30	42	48	48
Variables	128	128	160	110	43	21	126	96	81	33
Rows	81	81	176	126	69	25	139	85	129	81
Coefficients	384	384	520	420	159	86	488	368	353	161
Subproblems solved	16759	8959	355	115	941	556	93	13	1379	193
Total iterations	127743	103905	1431	663	2241	722	896	180	7952	481
Max. depth	41	41	20	13	20	17	18	5	32	17
CPU time	15244	15242	511	346	272	239	150	72	1223	257

The sizes and solution times for the two formulations are given in Table 2. This problem exemplifies the fact that it is invariably simpler to formulate such problems in disjunctive form and that this form is more explicable to the layman. Linearization of disjunctive constraints by introducing logical variables is a tedious and error prone process.

6. Implementation

The conventional and disjunctive algorithms were compared by coding (in FORTRAN 77 on a VAX/750) the Land–Doig algorithm [8] with MINOS 4.0 [10] being used to solve subproblems. In particular, logical relationships amongst constraints can be defined in the MPS file, and surrogates are automatically calculated and inserted into the matrix.

The major difficulty of the implementation is the need to modify the matrix. For example, a disjunction named ROW1OR2 comprising terms ROW1 and ROW2 would be defined in the ROWS section as follows:

```
OR ROW1OR2  ROW1  ROW2
L  ROW1
G  ROW2
```

When coefficients for ROW1 or ROW2 are read in from the COLUMNS and RHS sections, they are not put into the matrix, but stored elsewhere. When the MPS file has been read, the variables' bounds are available and the surrogate of ROW1OR2 is calculated. The coefficients of this surrogate are now inserted into the matrix, and the relaxed problem can be solved.

If arbitration entails replacing ROW1OR2 by say ROW1, the coefficients of the former are all set to zero, and some or all overwritten by the coefficients of ROW1. Relaxation is similar. This manipulation is inconvenient and expensive, because the matrix is stored in column order. Quite frequently however, a term of a disjunction is a bound on a variable, and arbitration or relaxation is effected more economically by modifying that bound. When a disjunction is a Generalized Binary Variable, relaxation and arbitration are performed wholly by manipulating bound intervals, and no matrix rows are required.

The conventional implementation chooses the logical variable whose value in the relaxation is closest to 0.5. This variable is first arbitrated upwards.

7. Results

The results shown in Table 2 were obtained for five problems solved with both the disjunctive and conventional algorithms, the surrogates of sections 2.1 and 2.2 being used throughout. The CPU time was measured in seconds on a VAX 750. The problems were:

1. A mine planning problem from [15, p. 247] somewhat enlarged.

2. A randomly generated job scheduling problem with four products on four machines.

3. A randomly generated 8*8 transport problem where each link has a finite capacity and a fixed cost component. In terms of finding the optimal feasible solution, the disjunctive algorithm performed badly on this problem. However, the first solution obtained by the disjunctive algorithm was much better than the first solution obtained by the conventional algorithm. Unlike the other test problems, neither the number of rows nor the number of variables was reduced by a disjunctive formulation.

4. A blending problem [15, p. 24] with a modification. Many of the surrogates used in this problem were bound intervals.

5. The bus scheduling problem outlined above.

Variations of problems 1 and 3 of different sizes were solved. The times thus obtained were consistent with the results presented here.

8. Conclusion

The disjunctive algorithm (DA) seems to have the following advantages over the conventional algorithm (CA).

● It is easier and more natural to express logical relationships amongst constraints directly than by expressing them in terms of logical variables, a difficult to automate and error prone process. A disjunctive formulation is often smaller than a conventional formulation. This is especially so when a disjunction comprises bounds on a single variable.

● It is clear that in some cases the disjunctive algorithm gives faster solutions than a parallel implementation of the conventional algorithm. This is probably due to the generally smaller size of the relaxed problem and better selection of entities to be arbitrated. The relation between the size of an LP and its solution time is problematical (see [9] for an analysis and further references). Chvatal [4, p. 113] implies that, if the number of iterations of the revised simplex method is proportional to m, the number of rows, then the mean solution time would be approximated by $am^2 + bmz$, where z is the number of non-zeros in the matrix, and a, b are constants. Four of the five test problems had disjunctive relaxations which were appreciably smaller and more quickly solved then the conventional relaxations.

● The (DA) gives better and cheaper first solutions of some problems (see Table 3).

● The (DA) makes it possible to incorporate a variety of surrogates which may give tighter relaxations than the (CA). It also suggests some extensions which would be difficult to implement in the conventional framework.

The disadvantages of the disjunctive algorithm are:

● In general, subproblems are formed by altering the matrix. This implies that each subproblem must start with a reinversion. If, as is common, the terms of a disjunction are bounds, arbitration can be (and is) expressed by altering bounds on variables in which case reinversion is not needed. It should be noted that all subproblems in both computer programs start with a reinversion.

Table 3
Resources required to obtain first solutions

Problem	Algorithm	Nodes required for first solution	Value of first solution	Nodes required for full solution	Optimal disjunctive solution
Transport	Conventional	42	2551	16759	2020
	Disjunctive	46	2157	8959	
Mining	Conventional	13	295.3	355	332.9
	Disjunctive	13	262.9	115	
Bus	Conventional	16	188.3	941	186.75
	Disjunctive	18	191.00	556	
Blending	Conventional	17	119700	55	119700
	Disjunctive	5	119700	13	
Schedule	Conventional	569	41.00	1379	36.00
	Disjunctive	15	40.00	193	

● Generating, storing and altering the constraints is much more difficult and time consuming than retaining the values of arbitrated variables.

References

[1] Balas, E., "Disjunctive programming", in: Hammer, P.L., E.L. Johnson and B.H. Korte (eds.), *Discrete Optimization II, Annals of Discrete Mathematics 5*, North-Holland, Amsterdam, 1979.

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

[3] Beaumont, N.B., "A generalization of binary variables", Working paper of the Department of Information Systems, Monash University, 1988.

[4] Chvâtal, V., *Linear Programming*, W.H. Freeman and Company, New York, 1983.

[5] Daellenbach, H.G., and George, J.A., *Introduction to Operations Research Techniques*, Allyn and Bacon Inc., Boston, MA, 1978.

[6] Geoffrion, A.M., and Marsten, R.E., "Integer programming algorithms: A framework and state-of-the-art survey", *Management Science* 18 (1972) 465–491.

[7] Jeroslow, R.G., and Lowe, J.K., "Modelling with integer variables", *Mathematical Programming Studies* 22 (1984) 167–184.

[8] Land, A.H., and Powell, S., *Fortran Codes for Mathematical Programming*, Wiley, 1973.

[9] Megiddo, N., "Improved asymptotic analysis of the average number of steps performed by the self dual simplex algorithm", *Mathematical Programming* 35 (1986) 140–172.

[10] Murtagh, B.A., and Saunders, M.A., "MINOS user's guide", Technical report SOL 77-9, Department of Operations Research, Stanford University, CA, 1977.

[11] Rado, F., "Linear programming with logical conditions", *Matekon* 3 (1966) 52–56.

[12] Scicon Limited, *SCICONIC/VM User Guide*, Sciconic Limited, Milton Keynes, 1984.

[13] Sherali, H.D., and Sen, S., "On generating cutting planes from combinatorial disjunctions", *Operations Research* 33 (1985) 928–933.

[14] Sherali, H.D., and Shetty, C.M., *Optimization with Disjunctive Constraints*, Springer-Verlag, 1980.

[15] Williams, H.P., *Model Building in Mathematical Programming*, Wiley, 1985.