J.-F. Raimond

Minimaximal Paths in Disjunctive Graphs by Direct Search

Abstract: The problem of finding a minimaximal path in a disjunctive network is stated in terms of both graph theory and linear programming with mixed-integer variables. It is solved in both formulations using a "direct search" scheme with additional dynamic features, which seems to be a more efficient algorithm than those based on other methods. Although it yields an optimal solution, the algorithm can be used as such or with very few changes to find suboptimal solutions for larger problems. Computational experience on the general machine scheduling problem is described.

Introduction

A number of practical problems can be formulated in terms of graph theory by introducing disjunctive graphs, for example, the q-machine scheduling problem, which can be stated as follows:

The manufacturing of m items or m lots of items requires that each item be processed on some of q machines. The processing of one item by one machine is called an operation. The sequence of operations related to an item is fixed by the technological process. However, the sequence of operations associated with a machine is not fixed. The problem is to determine the q sequences of operations so that the total time to process the m items on the q machines is a minimum.

A node represents an operation and nodes 0 and n represent the dummy operations of beginning and finishing the process. Then an ordinary oriented arc relates two nodes corresponding to two consecutive operations on one item; a disjunctive arc joins any two nodes corresponding to operations occurring on the same machine. Let us define these terms.

A "disjunctive arc" joining nodes i and h is an arc which has, a priori, neither an orientation nor a length, but is associated with two positive numbers d_i and d_h. Such an arc is represented by the symbolism in Fig. 1. If we choose orientation from i to h, this arc becomes an ordinary oriented arc of length d_i; if we choose the opposite orientation, the length is d_h. We call "selection" the choice of one orientation of a disjunctive arc.

The author is located at Compagnie IBM France, 47, rue de Villiers, 92-Neuilly-sur-Seine, France.

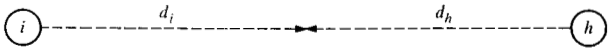

Figure 1 Disjunctive arc.

A "disjunctive graph" G is a graph $G(X, A, B)$ in which X is a set of nodes, A is a set of disjunctive arcs and B is a set of ordinary oriented (or conjunctive) arcs. Without loss of generality, we assume that all arcs leaving node i have the same length and that there exist two nodes, indexed 0 and n, neither of which is an extremity of a disjunctive arc. An example of a disjunctive graph is shown in Fig. 2.

Let w be a complete set of selections over A. We obtain from the disjunctive graph in Fig. 2 an ordinary oriented graph that we denote by $G(w)$, Fig. 3. On $G(w)$ two situations are possible:

1. There is no loop. Then there exists at least one longest or critical path joining node 0 to node n. Let η_w be the length of a critical path in $G(w)$.
2. There is at least one loop. Then there exists no critical path of finite length in $G(w)$. We denote by η'_w the length of one loop in $G(w)$.

The problem we deal with in this paper consists of finding a complete set of selections w^* such that there is no loop in $G(w^*)$ and the length η_w of a critical path in $G(w^*)$ is a minimum over all possible complete sets of selections. Such a path is therefore called a "minimaximal" path in G.

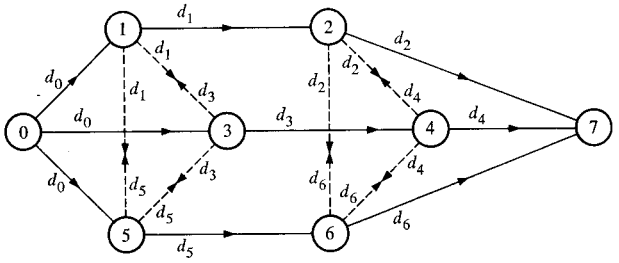

Figure 2 Disjunctive graph.

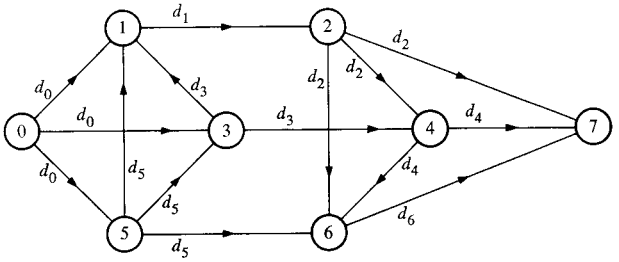

Figure 3 Complete set of selections from the disjunctive graph in Fig. 2.

This problem has been studied by numerous authors[1-10] in connection with the machine scheduling problem. Roy and Sussmann[8,11,12] and Greenberg[10] use a branch-and-bound algorithm, while Balas[1,13,14] uses Benders' partitioning algorithm.[15]

The purpose of this paper is to present a new method of solving the problem using the "direct search" technique of Lemke and Spielberg[16-18] for mixed-integer, zero-one linear programming. The feasibility of this method was demonstrated in a preliminary report.[19] Let us briefly review the method; the problem solved is the following:

$$\text{MP}\begin{cases}\min z = c^{\mathrm{T}}x + q^{\mathrm{T}}y, \\ Dx + Ey \leq b, \\ x \geq 0 \quad \text{and} \\ y \text{ is a } (0, 1) \text{ vector.}\dagger\end{cases}$$

All vectors are column vectors and the superscript T indicates transposition. If we fix y at some value y^k, the problem becomes

$$\text{LP}\begin{cases}\min Z = c^{\mathrm{T}}x, \\ Dx \leq b - Ey^k \quad \text{and} \\ x \geq 0.\end{cases}$$

LP is a classical linear programming problem. Its dual is

$$\text{DP}\begin{cases}\max \zeta = (Ey^k - b)^{\mathrm{T}}u, \\ D^{\mathrm{T}}u \geq -c \quad \text{and} \\ u \geq 0.\end{cases}$$

There is only a finite number of different y vectors. The algorithm uses a search over the y's, starting with $y^0 = 0$. The set of all possible values of y can be represented by a graph Y in which each node R^k represents one value y^k of y and $R^{k'}$ is a "successor" of R^k if and only if $y^{k'}$ can be derived from y^k by setting exactly one of the 0 components of y^k to 1. A Y graph drawn for a four-dimensional y vector is shown in Fig. 4. The search is made by scanning this graph in such a way that one never comes twice on a forward step to the same node. (The procedure also has features that allow sets of nodes to be eliminated from the search.) A forward step consists of going from the currently scanned node to one of its successors; a backward step is a move to the predecessor node from which the currently scanned node was reached.

At each scanned node the dual problem is solved and yields either an optimal solution u^k or the direction vector v^k of an extreme ray of the cone associated with the solution set. Let z^* be an upper bound of the objective function (practically, we take the best value of the objective function found so far). Then the following constraints on y are generated:

If LP is feasible,

$$(q + E^{\mathrm{T}}u^k)^{\mathrm{T}}y - u^{k\mathrm{T}}b \leq z^*. \tag{1}$$

If LP is not feasible,

$$v^{k\mathrm{T}}(Ey - b) \leq 0. \tag{2}$$

Although these constraints are global (i.e., true for every y), one uses only the constraints generated at the current node to determine a preferred set of variables (i.e., a subset of all possible branches emanating from the current node, which exhausts the branches that must be taken) and possibly a rule of choice among the preferred variables. If the preferred set is empty, then a backward step is initiated. These constraints can be modified in their coefficients to yield local constraints, i.e., constraints which are valid only for successors of the current node. Because only the current constraint is used, the same computations as for the global constraint can be made on this local constraint, which is generally stronger than the global one.

In succeeding sections we first formalize the problem as a mixed-integer linear programming problem, then we show how the search is made and finally we describe the features used at each step of the algorithm.

† The value of each of the components of the vector y is either zero or one.

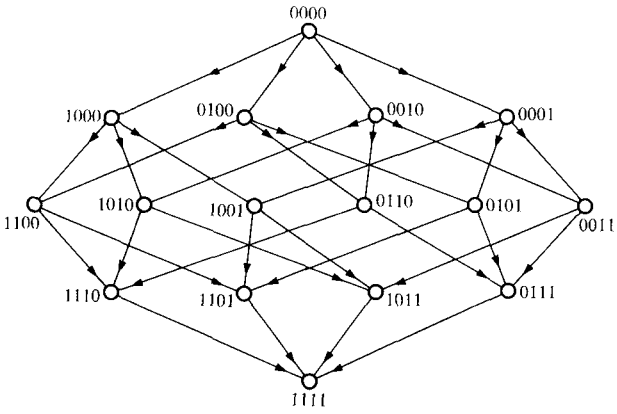

Figure 4 Graph Y for a four-dimensional y vector.

Mixed-integer formulation

Let $p = |A|$. Then we identify any complete set of selections w by a p-dimensional zero-one vector y^w in the following way: We index all the disjunctive arcs of G in an arbitrary manner by $j \in A = \{1, 2, \cdots, p\}$. Furthermore, we choose an arbitrary initial complete set of selections Ω, identified by y^Ω such that $y_j^\Omega = 0$, $j \in A$. Any complete set of selections w will be identified by y^w such that

$y_j^w = 0$ if the selection on arc j is the same in w and in Ω or

$y_j^w = 1$ if the selections on arc j in w and in Ω are different.

This establishes a one-to-one correspondence between the w's and the y^w's.

For a given $G(w)$ we call "normal" arcs those arcs for which $y_j^w = 0$ and "inverse" arcs those arcs for which $y_j^w = 1$. Throughout this paper the index i always stands for the origin node of the normal arc j and h stands for its endpoint; i and h are thus functions of j and arc (i, h) is arc j.

Now we attach to any node of $G(w)$ a continuous variable t_l, $l \in X$. For a given w the problem of finding the length of a critical path in $G(w)$ can be formulated as

$$\begin{cases} \min Z = t_n; & (3) \\ t_m - t_l \geq d_l, \qquad (l, m) \in B; & (4) \\ t_h - t_i \geq d_i, \ (i, h) = j \in A \cap \{j \mid y_j^w = 0\}; & (5) \\ t_i - t_h \geq d_h, \ (i, h) = j \in A \cap \{j \mid y_j^w = 1\}. & (6) \end{cases}$$

Let Δ_j be an upper bound of $|t_h - t_i|$ for every w and let $\delta_j^0 = \Delta_j + d_i$ and $\delta_j^1 = \Delta_j + d_h$. Then constraints (3) and (4) are equivalent to

$$t_h - t_i - d_i \geq -y_j^w \delta_j^0 \quad \text{and} \tag{7}$$

$$t_i - t_h - d_h \geq -(1 - y_j^w)\delta_j^1, \qquad j \in A. \tag{8}$$

Constraints (7) and (8) can be established by noting that, if $y_j^w = 0$, then (7) is the same as (5) and (8) becomes

$$t_h - t_i + d_h \leq \Delta_j + d_h.$$

However, (5) implies $t_h - t_i > 0$; therefore $|t_h - t_i| = t_h - t_i$ and (8) is certainly verified. Similar reasoning applies to the case $y_j^w = 1$.

Now the mixed-integer formulation of the problem is

$$\begin{cases} \min Z = t_n; & (3) \\ t_l - t_m \leq -d_l, \qquad (l, m) \in B; & (9) \\ t_i - t_h - \delta_j^0 y_j \leq -d_i, \qquad j \in A; & (10) \\ t_h - t_i + \delta_j^1 y_j \leq \delta_j^1 - d_h, \quad j \in A; & (11) \\ t_i \geq 0, \qquad i \in X; \quad \text{and} & (12) \\ y_j = 0 \quad \text{or} \quad 1, \qquad j \in A. & (13) \end{cases}$$

If we fix y at some value y^w, we obtain a linear problem which is a PERT (program evaluation review technique) problem on the graph $G(w)$ (with some redundant constraints). The following basic results should be remembered:

1. The t_i's form a system of potentials on the graph.
2. The formulation of this problem with a conjunctive system of constraints yields a linear program in terms of the t_i's, the number of constraints equaling the number of arcs in the graph, and the value of the objective function at the optimum is the length of a critical path.
3. In the dual problem each variable u_j is attached to an arc. If the network has no loop, an optimal solution is given by $u_j = 1$ if arc j lies on the critical path, or by $u_j = 0$ if not. If the network has loops, extreme rays are given for each loop by $u_j = 1$ if arc j lies on the loop, or by $u_j = 0$ if not.

Organization of the search

Consider graph Y which has 2^p nodes. We call "level k" the subset of nodes of Y for which exactly $k - 1$ components of y are 1 ($k = 1, 2, \cdots, p + 1$). In Fig. 4 nodes of a given level lie on the same horizontal line. All direct successors of a node on level k lie on level $k + 1$, while the predecessors lie on level $k - 1$. At each iteration the indices j of the components of y (or the variables y_j) are divided into three sets which effect a partition of A:

F is the set of free indices (or variables); free variables are currently 0.

H^0 is the set of indices (or variables) such that y_j is fixed at 0.

H^1 is the set of indices (or variables) such that y_j is fixed at 1.

An exhaustive search is realized by the following scheme: Let k be the current level. A forward step consists of branching to a free direct successor of the current node, i.e., we transfer some index j_0 from F to H^1. A backward step consists of branching to the predecessor from which the current node was reached and of forbidding access to the current node. This is accomplished by transferring from H^1 to H^0 the index j that was transferred earlier to H^1 when branching to the current node. Further, all indices brought into H^0 by backward steps between levels $k+1$ and k or canceled (see *Cancellation test*) at level k are freed (i.e., transferred from H^0 to F). Note that all variables which have the value 1 at some node R keep this value for all successors of R, i.e.,

$$S \in \Gamma^*(R) \Rightarrow H_R^1 \subseteq H_S^1.$$

We also define $\bar{H}^1 = H^0 \cup F$.

For any graph $G(w)$ without loops, we denote by γ_w the subset of indices $j \in A$ such that the disjunctive arc j kept in $G(w)$ is an element of the critical path Z_w. Also, for a graph $G(w)$ with loops, we call γ_w' the subset of indices $j \in A$ such that the disjunctive arc j kept in $G(w)$ is an element of a loop of length η_w'. From a computational point of view the state of the search is described by two p-dimensional vectors:

φ is a sequence vector that contains in sequence the k indices $j \in H^1$.

ψ is a state vector defined as
- $\psi_j = 0 \Leftrightarrow j \in F$,
- $\psi_j = k \Leftrightarrow j$ has been fixed to 1 at level k or
- $\psi_j = -k \Leftrightarrow j$ has been fixed to 0 at level k.

Note that there are one-to-one correspondences among a complete set of selections w, the particular value y^w of y and the corresponding node R^w of Y, which allow us to use any of these symbols (words) in place of the others.

Basic elements of the algorithm

- *Preferred set*

At the current node in the search let w be the corresponding complete set of selections and let Z_w be a critical path of $G(w)$. For every complete set of selections that has the same selections as w on γ_w, the critical path cannot be shorter than Z_w because the critical path is the longest path between nodes 0 and n in $G(w)$. Therefore, one needs to select only branches that correspond to an inversion of some disjunctive arc on Z_w. The preferred set is therefore

$$\Pi = F \cap \gamma_w. \tag{14}$$

In the same way, if $G(w)$ has a loop, then some disjunctive arc on the loop must be reversed and therefore

$$\Pi = F \cap \gamma_w'.$$

If $\Pi = \emptyset$ (the empty set), a backward step is taken.

- *Derivation of inequalities*

We now derive the inequalities (1) and (2) for this problem. Note that in our case $q = 0$. Therefore the inequalities have the same coefficient on the left-hand side:

$(Ey - b)^T u^k \leq z^*$ if feasible or

$(Ey - b)^T v^k \leq 0$ if not feasible.

We describe only the computations related to the first case. Computations for the second one differ only by the right-hand side of the corresponding equations. Let

$$u^T = (u^{2T}, u^{0T}, u^{1T}),$$

where u^2, u^0 and u^1 are the dual vectors associated with constraints (9), (10) and (11), respectively; u^0 and u^1 are p-dimensional. Let R^w be the current node and y^w the attached value of y.

The objective function of the dual problem is

$$\zeta = (Ey^w - b)^T u,$$

where

$$Ey^w - b = \begin{bmatrix} d_\mu \\ d_i - \delta_i^0 y_i^w \\ d_h - \delta_i^1(1 - y_i^w) \end{bmatrix}.$$

Let $\mu \in X$ be the origin of the ordinary arc corresponding to the dual variable u_i^2. Then

$$\zeta = \sum_{l \in B} d_\mu u_l^2 + \sum_{j \in A} (d_i - \delta_i^0 y_i^w) u_i^0 + \sum_{j \in A} [d_h - \delta_i^1(1 - y_i^w)] u_i^1.$$

Let $\bar{u}$ be an optimal solution. The dual problem maximizes ζ. Therefore, since $u \geq 0$, we have the following implications:

$$j \in H^1 \Rightarrow y_j^w = 1 \Rightarrow d_i - \delta_i^0 y_i^w < 0 \Rightarrow \bar{u}_i^0 = 0 \text{ and}$$

$$j \in \bar{H}^1 \Rightarrow y_j^w = 0 \Rightarrow d_h - \delta_i^1(1 - y_i^w) < 0 \Rightarrow \bar{u}_i^1 = 0.$$

Therefore the problem is, equivalently, to maximize

$$\zeta^1 = \sum_{l \in B} d_\mu u_l^2 + \sum_{i \in \bar{H}^1} d_i u_i^0 + \sum_{j \in H^1} d_h u_i^1.$$

However, this is the objective function of the dual PERT problem on the conjunctive graph $G(w)$. Moreover, the remaining constraints

$$D^T u \geq -c,$$

$$u \geq 0,$$

$u_j^0 = 0, \quad j \in H^1$ and

$u_j^1 = 0, \quad j \in \bar{H}^1,$

are the constraints of the classical PERT problem on $G(w)$. The remaining components of $\bar{u}$ are 1 for those arcs that are elements of a critical path Z_w and 0 otherwise. Specifically,

$\bar{u}_l^2 = 1$ on purely conjunctive arcs $\in Z_w$,

$\bar{u}_j^0 = 1, \quad j \in \bar{H}^1 \bigcap \gamma_w$ and

$\bar{u}_j^1 = 1, \quad j \in H^1 \bigcap \gamma_w.$

The constraint generated at the current node is

$$(Ey - b)^T \bar{u} \leq z^*,$$

which can be written in expanded form as

$$\sum_{l \in B} d_\mu \bar{u}_l^2 + \sum_{j \in A} (d_i - \delta_i^0 y_i)\bar{u}_i^0 + \sum_{j \in A} [d_h - \delta_i^1(1 - y_i)]\bar{u}_i^1 \leq z^*.$$

Substituting the values of $\bar{u}_i^0$ and $\bar{u}_i^1$, one gets

$$\sum_{l \in B} d_\mu \bar{u}_l^2 + \sum_{j \in \bar{J}} (d_i - \delta_i^0 y_i) + \sum_{j \in J} [d_h - \delta_i^1(1 - y_i)] \leq z^*,$$

where $\bar{J} = \bar{H}^1 \bigcap \gamma_w$ and $J = H^1 \bigcap \gamma_w$. Now, recalling that the length η_w of the critical path Z_w is

$$\eta_w = \sum_{l \in B} d_\mu \bar{u}_l^2 + \sum_{j \in \bar{J}} d_i + \sum_{j \in J} d_h,$$

one gets the final form of the global constraint:

$$-\sum_{j \in \bar{J}} \delta_i^0 y_i - \sum_{j \in J} \delta_i^1(1 - y_i) \leq z^* - \eta_w. \tag{15}$$

However, because this inequality will be used only at the current node, we are more interested in the local inequality, i.e., an inequality which is true for all successors of the current node. For every value of ν such that the corresponding node is a successor of the current node, we have the implications

$j \in H^1 \Rightarrow y_j^\nu = 1$ and

$j \in H^0 \Rightarrow y_j^\nu = 0.$

Therefore (15) yields the following local constraint for the feasible case:

$$-\sum_{j \in \Pi} \delta_i^0 y_i \leq z^* - \eta_w. \tag{16}$$

If there are loops, the same computations yield

$$-\sum_{j \in \Pi'} \delta_i^0 y_i \leq -\eta_w', \tag{17}$$

where $\Pi' = F \bigcap \gamma_w'$.

While it is possible to generate as many inequalities of type (16) or (17) as there are critical paths or loops in $G(w)$, we consider only one inequality at each node. Also, for simplicity, we let $-\xi_w$ generically represent the right-hand side of (16) or (17) and deal with

$$-\sum_{j \in \Pi} \delta_i^0 y_i \leq -\xi_w. \tag{18}$$

Constraint (18) is used in two ways:

Ceiling test

The best way to satisfy (18) is to set every y_i, $j \in \Pi$, equal to 1. Therefore, if

$$\sum_{j \in \Pi} \delta_i^0 \leq \xi_w, \tag{19}$$

a backward step is taken.

Reduction of the inequality

By combining (18) with the inequalities

$$y_i \leq 1, \quad j \in \Pi, \tag{20}$$

one can obtain the inequality

$$-\sum_{j \in \Pi_w^0} \delta_i^0 y_i \leq -\xi_w^0. \tag{21}$$

Any further combination of (21) with (20) to reduce Π_w^0 would yield a negative value of ξ_w^0.

The search is continued by setting to 1 some variable y_i, where $j \in \Pi_w^0$. When this set has been exhausted, it is necessary to start again from (18) if (19) is currently satisfied. Otherwise, a backward step is taken. For example, using (18) one has $-(y_1 + 2y_2 + 5y_3) \leq -4$, $\Pi_w^0 = \{3\}$. When y_3 is set to 0, the constraint becomes $(y_1 + 2y_2) \geq 4$, which cannot be satisfied, and a backward step is taken.

• *Lower bound and further backtracking tests*

We introduce now, for a node corresponding to w, the partial set of selections σ defined as follows: σ contains only selections for $j \in H^0 \bigcup H^1$ and these are the same selections as are in w. Also, we define as $\Gamma(\sigma)$ the graph which is derived from $G(w)$ by making the selection σ and by deleting the disjunctive arcs that are not selected in σ. Such a graph is shown in Fig. 5.

In terms of $\Gamma(\sigma)$, two situations are possible:

1. $\Gamma(\sigma)$ has loops; this means that $G(w)$ has the same loops. Therefore, there exists a set γ_w' such that $F \bigcap \gamma_w' = \emptyset$ and a backward step must be taken. [Note that this test is necessary because the loop selected in solving the PERT problem on $G(w)$ may be different from the loops of $\Gamma(\sigma)$ and contain some index $j \in F$.]

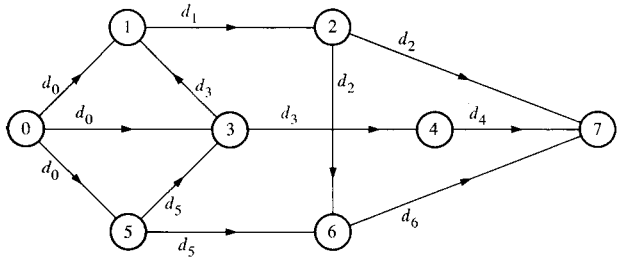

Figure 5 Graph $\Gamma(\sigma)$ derived from the graph in Fig. 3.

2. $\Gamma(\sigma)$ has no loops; then the PERT problem on $\Gamma(\sigma)$ is feasible and yields a critical path Z_σ of length η_σ. Any graph $G(\nu)$ corresponding to a successor of the current node is obtained by adding arcs to $\Gamma(\sigma)$. Therefore $\eta_\sigma \leq \eta_\nu$ and η_σ is a lower bound of the objective function for all successors of the current node. If

$$\eta_\sigma \geq z^*, \tag{22}$$

a backward step is taken.

Case 2 contains the case in which, solving the PERT problem on $G(w)$, one would have selected a critical path Z_w with $F \cap \gamma_w \neq \emptyset$, although there exists in $G(w)$ another critical path Z''_w with $F \cap \gamma''_w = \emptyset$.

• *Dynamic evaluation of upper bounds*

The efficiency of the algorithm depends partly on finding the lowest possible values of the upper bounds δ_j. We compute local values of δ_j at each step of the algorithm.

Inequality (18) involves only δ_j^0. Considering constraints (10) and (11), we see that the value of δ_j^0 has meaning only when $y_i = 1$, i.e., when one has selected the inverse arc of pair j. The constraint to be enforced is then

$$t_i - t_h - d_h \geq 0,$$

which implies $t_i - t_h > 0$. We want condition (10) to be satisfied automatically if (11) is satisfied. To ensure this, it is sufficient to take

$$\delta_j^0 \geq t_i - t_h + d_i,$$

i.e., to have $\Delta_j \geq t_i - t_h$ but not $\Delta_j \geq |t_i - t_h|$.

Let α_i^σ (β_i^σ) be the length of the largest path joining node 0 (i) to node i (n). Consider any value of ν corresponding to a successor of the current node such that $G(\nu)$ has no loop. Because $G(\nu)$ can be derived from $\Gamma(\sigma)$ by the addition of some arcs, the following inequalities hold:

$$t_h^\nu \geq \alpha_h^\sigma \quad \text{and}$$

$$\eta_\nu - t_i^\nu \geq \beta_i^\sigma.$$

Therefore

$$t_i^\nu - t_h^\nu \leq \eta_\nu - \alpha_h^\sigma - \beta_i^\sigma.$$

Furthermore, we are interested only in finding better solutions. Therefore we can enforce the additional constraint $\eta_\nu < z^*$ and obtain

$$t_i^\nu - t_h^\nu < z^* - \alpha_h^\sigma - \beta_i^\sigma. \tag{23}$$

Inequality (23) is true for every successor of the current node; its use yields a solution better than the best one found so far. Therefore, in the local inequality (18) we use values of δ_j^0 given by

$$\delta_j^0 = z^* - \alpha_h^\sigma - \beta_i^\sigma + d_i. \tag{24}$$

• *Cancellation test*

Combining inequalities (23) and (11) with $y_i = 1$, one gets the inequality

$$d_h < z^* - \alpha_h^\sigma - \beta_i^\sigma. \tag{25}$$

If this inequality is not satisfied, one will not get a better solution by setting y_i to 1. For any successor (ν, τ) of the current node corresponding to a complete set of selections ν and a partial set τ, one has

$$\alpha_h^\tau \geq \alpha_h^\sigma,$$

$$\beta_i^\tau \geq \beta_i^\sigma \quad \text{and}$$

$$z'^* \leq z^*,$$

which can be combined as

$$z'^* - \alpha_h^\tau - \beta_i^\tau \leq z^* - \alpha_h^\sigma - \beta_i^\sigma. \tag{26}$$

Therefore if (25) is not satisfied at the current node, it will not be satisfied at any successor node. Thus one has the following cancellation test: If

$$\delta_j^0 \leq d_i + d_h, \tag{27}$$

cancel variable j, i.e., transfer j from F to H^0. Note that this is not a backtracking test; all indices such that (27) is not verified remain free.

The flow diagram for the algorithm is shown in Fig. 6.

Additional features of the algorithm

• *Choice of the initial complete set of selections*

The origin of the search is fixed by the arbitrary choice of the initial complete set of selections Ω. However, if the origin is a good solution one can expect that the optimal solution will be obtained with a relatively small number of iterations. These considerations led us to use a heuristic procedure to derive the initial set of selections.

Let λ be the empty set, i.e., $\Gamma(\lambda)$ is the graph obtained from $G(w)$ by deleting all disjunctive arcs. Let r and s be any two nodes related by a disjunctive arc. It is evident

that the greater the quantity $(\eta_\lambda - \beta_s^\lambda - \alpha_r^\lambda - d_r)$ is, the more likely it is that in the optimal graph the disjunctive arc will be oriented from r to s. This observation leads to the following rule for the construction of the initial complete set of selections: If

$$\eta_\lambda - \beta_s^\lambda - \alpha_r^\lambda - d_r > \eta_\lambda - \beta_r^\lambda - \alpha_s^\lambda - d_s, \quad \text{i.e., if}$$

$$\beta_r^\lambda + \alpha_s^\lambda + d_s > \beta_s^\lambda + \alpha_r^\lambda + d_r,$$

choose the orientation from r to s; otherwise, choose the orientation from s to r.

• *Strategies*

We call a strategy the set of rules used to determine which element of the preferred set is to be selected for the next branch after the preferred set has been reduced as much as possible. A first strategy is to try to get good solutions quickly; then one can stop the algorithm after some fixed execution time. One has a set of good solutions, but optimality is not proved. We call such a strategy, in which we branch to the node that hopefully will give the best value of the objective function, a "next-best" strategy. However, when one has a good solution, he may be interested in ascertaining optimality as quickly as possible. To do that, he might do well to cut branches after as few links as possible and so to use a "next-worst" strategy.

Until now, the only data we have to evaluate the quality of a step are inequalities (18) or (27). A next-best choice will be one that maximally reduces infeasibility. Therefore two strategies are possible:

1. From (18): Select k such that

$$\delta_k^0 = \max_{j \in \Pi} \delta_j^0. \tag{28}$$

2. From (27): Select k such that

$$\delta_k^0 - d_{i(k)} - d_{h(k)} = \max_{j \in \Pi} (\delta_j^0 - d_i - d_h). \tag{29}$$

Next-worst strategies would minimize these quantities.

• *Relocation of the origin*

At some point in the search, either fixed a priori by some maximum number of iterations or under the control of the operator if some information is displayed visually, stop the search, select as the initial Ω the complete set of selections w^* yielding the best solution found so far and restart the algorithm. This process, called relocation of the origin,[18,20,21] can be handled without modifying the algorithm because, in fact, there is no difference between a normal arc and an inverse arc. One uses a new vector $\tilde{y}$ defined by

$$\tilde{y}_j = y_j \quad \text{if} \quad y_j^* = 0 \quad \text{or}$$

$$\tilde{y}_j = 1 - y_j \quad \text{if} \quad y_j^* = 1.$$

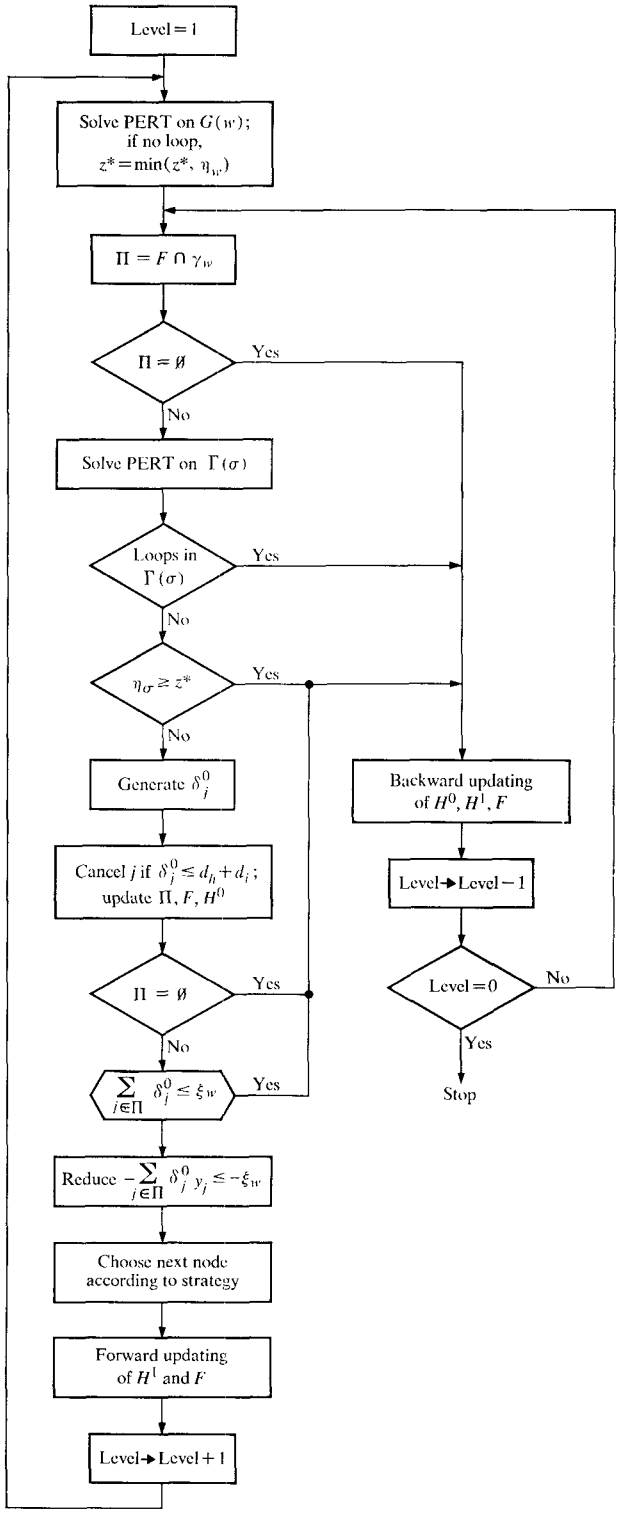

Figure 6 Flow diagram for the algorithm.

This means that the new initial complete set of selections is obtained from the old one by reverting the arcs corresponding to $y_j^* = 1$. In other words, the only change is to define new values $\tilde{l}$ and $\tilde{h}$ by

Table 1 Summary of computational experience with the machine scheduling problem.

Run	*Number of items*	*Number of machines*	*Number of disjunctive arcs*	*Initial choice of w?*	*Number of iterations*	*Number of solutions*	*First solution value*	*Best solution value*	*Optimality proved?*
1	2	3	3	yes	1	1	13	13	yes
2	3	2	6	yes	3	2	10	8	yes
3	3	2	6	no	5	2	34	31	yes
4	5	4	15	no	34	9	24	13	yes
5	5	4	15	yes	13	3	15	13	yes
6	5	5	28	yes	15	3	73	66	yes
7	7	4	66	no	36	9	153	98	yes
8	7	4	66	yes	1	1	98	98	yes
9	10	4	153	no	10,000	19	328	159	no
10	10	4	153	yes	9,600	8	168	153	no

$\tilde{i} = i, \quad \tilde{h} = h \quad$ if $\quad y_i^* = 0 \quad$ or

$\tilde{i} = h, \quad \tilde{h} = i \quad$ if $\quad y_i^* = 1.$

Then the algorithm is restarted using the new values $\tilde{i}$ and $\tilde{h}$.

• *Heuristics*

If one is interested only in getting good solutions, or if the problem is too large, one can use heuristic procedures to reduce computation time. Two techniques are particularly easy to implement, but they are by no means the only ones that could be used:

1. Fix, a priori, some selections. Choose Ω so that it is compatible with these selections and start the algorithm with $H^0 \neq \emptyset$.
2. Use a tolerance T; in all tests involving z^* replace z^* by $z^* - T$. In general this is very efficient, but the value of the objective function for the best solution found differs from the optimal value by a quantity not larger than T.

Summary and computational experience

We have applied the direct search scheme to the problem of finding a minimaximal path in a disjunctive PERT network. The structure of the problem and its formulation as a mixed-integer linear program proved to be particularly relevant to this technique.

From a theoretical viewpoint, the main advantages of the method are the following:

1. Using specialized Benders' constraints[15] we obtain a backtracking test that can be used with lower and upper bounds.
2. These constraints also allow canceling some variables at each step.
3. Introducing the partial conjunctive graphs, we obtain dynamic evaluation of the upper bounds used in the mixed-integer formulation.

From a practical viewpoint, the use of the direct search has the following advantages:

1. It is the implicit enumerative scheme that requires the least storage for intermediate results.
2. No difficult problem has to be solved at each step, in contrast with methods based on Benders' partitioning algorithm.[15]
3. It is a primal feasible algorithm, i.e., it yields a sequence of feasible solutions. In practice, it is often more interesting to have several good solutions than only one optimal solution.
4. As a consequence of Point 3 we can set, a priori, a maximum processing time and use the best solution found.

A FORTRAN IV program implementing these ideas in terms of the machine scheduling problem was written for the IBM 360/40. The most significant results are summarized in Table 1. Runs 1 and 2 are related to sample problems 3 and 4 of Ref. 10. No comparison is possible because Greenberg does not give a result for the make-span problem. Runs 3, 4 and 5 are related to two examples of Balas.[1] Benders' partitioning scheme for these examples uses 11 and 40 iterations, respectively, each of which is longer than one iteration of our algorithm. The other runs are related to randomly generated problems. For large problems, use of the heuristic procedure for the initial choice of origin seems to be equivalent to relocating the origin after 100 iterations; that is, the best solution found in the first 100 iterations without benefit of the heuristically chosen origin is generally the one obtained by using the initial heuristic choice of origin. Use of a

tolerance also decreases the number of iterations needed to find the best solution. Runs 9 and 10, for which optimality was not proved, were stopped after nine minutes on an IBM 360/75.

Acknowledgments

I am very grateful to Kurt Spielberg for his continuous interest and valuable comments during the development of the algorithm and to Egon Balas for introducing me to the machine scheduling problem and for later discussions of various points of interest.

References

1. E. Balas, "Finding a Minimaximal Path in a Disjunctive PERT Network," *Théorie des Graphes, Journées Internationales d'Etudes, Rome, 1966,* Dunod, Paris 1967.
2. R. J. Giglio and H. M. Wagner, "Approximate Solution to the Three-Machine Scheduling Problem," *Operations Res.* **16,** 305, (1964).
3. E. Ignall and L. Schrage, "Application of the Branch-and-Bound Technique to Some Flow-Shop Scheduling Problems," *Operations Res.* **13,** 400 (1965).
4. S. Johnson, "Optimal Two- and Three-Stage Production Schedules with Setup Times Included," *Naval Res. Log. Quart.* **1,** 61 (1954).
5. Z. A. Lomnicki, "A Branch-and-Bound Algorithm for the Exact Solution of the Three-Machine Scheduling Problem," *Operational Res. Quart.* **16,** 89 (1965).
6. G. B. McMahon and P. G. Burton, "Flow-Shop Scheduling with the Branch-and-Bound Method," *Operations Res.* **15,** 473 (1967).
7. D. S. Palmer, "Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time—A Quick Method of Obtaining Near Optimum," *Operational Res. Quart.* **16,** 101 (1965).
8. B. Roy and B. Sussmann, "Les Problèmes d'Ordonnancement avec Contraintes Disjonctives," *Rapport de Recherches No. 9,* Société d'Economie et de Mathématiques Appliquées, Paris 1964.
9. H. M. Wagner, "An Integer Linear Programming Model for Machine Scheduling," *Naval Res. Log. Quart.* **6,** 131 (1959).
10. H. H. Greenberg, "A Branch-and-Bound Solution to the General Scheduling Problem," *Operations Res.* **16,** 353 (1968).
11. B. Roy, "Cheminement et Connexité dans les Graphes; Applications aux Problèmes d'Ordonnancement," *METRA Série Spéciale No. 1,* METRA International, Paris 1962.
12. P. Bertier and B. Roy, "Procédure de Résolution pour une Classe de Problèmes pouvant avoir un Caractère Combinatoire," *Cahiers du Centre d'Etudes de Recherche Opérationnelle (Bruxelles)* **6,** 202 (1964).
13. E. Balas, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Operations Res.* **13,** 517 (1965).
14. E. Balas, "Discrete Programming by the Filter Method," *Operations Res.* **15,** 915 (1967).
15. J. F. Benders, "Partitioning Procedures for Solving Mixed-Variable Programming Problems," *Numerische Mathematik* **4,** 238 (1962).
16. C. E. Lemke and K. Spielberg, "Direct Search Zero-One and Mixed-Integer Programming," *IBM New York Scientific Center Report 320–2911,* 1966.
17. C. E. Lemke and K. Spielberg, "Direct Search Algorithm for Zero-One and Mixed-Integer Programming," *Operations Res.* **15,** 892 (1967).
18. K. Spielberg, "An Algorithm for the Simple Plant Location Problem with Some Side Conditions," *IBM New York Scientific Center Report 320–2900,* 1967.
19. J.-F. Raimond, "An Algorithm for the Exact Solution of a Machine Scheduling Problem," *IBM New York Scientific Center Report 320–2930,* 1968.
20. K. Spielberg, "Plant Location with Generalized Search Origin," *IBM New York Scientific Center Report 320–2929,* 1967.
21. H. M. Salkin and K. Spielberg, "Adaptive Binary Programming," *IBM New York Scientific Center Report 320–2951,* 1968.

Received January 27, 1969