BA

stichting
mathematisch
centrum

MC

BA

AFDELING MATHEMATISCHE BESLISKUNDE BN 22/73 NOVEMBER

IN SAMENWERKING MET HET
INTERUNIVERSITAIR INSTITUUT BEDRIJFSKUNDE
DELFT/ROTTERDAM

J.K. LENSTRA & A.H.G. RINNOOY KAN

TOWARDS A BETTER ALGORITHM FOR THE JOB-SHOP SCHEDULING PROBLEM - I

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

AMS (MOS) subject classification scheme (1970): 90B35

Abstract

So far two fairly efficient branch-and-bound algorithms for the job-shop scheduling problem have been developed: one by Charlton and Death and one by Florian et al.. In this report we investigate the possibility of combining the good qualities of these algorithms into one new and hopefully more powerful approach. Though some questions remain unanswered, the enquiry seems worth pursuing.

Contents

1. Introduction

The *job-shop scheduling problem* can be formulated as follows:

> Given the order by which each of n *jobs* has to pass through m *machines*, and given the processing time of each *operation*, find the order by which each machine has to process the jobs so as to minimize the total processing time.

The problem is generally considered to be one of the most difficult sequencing problems and has been attacked rather unsuccesfully by many researchers. Recently, however, some *branch-and-bound algorithms* have been developed that seen to be able to handle at least a small number of jobs in an efficient way.

These algorithms are all based on the formulation of the problem by means of the highly useful concept of a *disjunctive graph* [1]. They differ, however, considerably with regard to the *branching strategy* used and the computation of a *lower bound*. We can roughly distinguish two groups of algorithms, typical examples of each group being given by the work of Charlton and Death [2,3] and Florian et al. [4,5], respectively. A description of these algorithms can be found in [6]; their basic principles will be summarized in section 2 below.

From this summary it will become apparent that ideally one should try to combine the superior branching strategy of Charlton and Death with the much stronger lower bound of Florian et al., as outlined in section 3.

We are at the moment engaged in an effort to effect this happy liaison. The object of this preliminary paper is to describe some of the problems encountered and the conclusions reached so far. Section 4 is devoted to this matter. Obviously, a lot of work remains to be done, but the present enquiry seems worth pursuing.

2. Summary of previous algorithms

First, we introduce some notation. We denote the n *jobs* by $J_1,\dots,J_n$ and the m *machines* by $M_1,\dots,M_m$. A job consists of a number of *operations*, each performed on a specific machine. The *disjunctive graph* is characterized by three sets $\mathcal{V}$, $\mathcal{C}$ and $\mathcal{D}$:

- $\mathcal{V}$ is the set of *vertices* of the graph; there is a vertex corresponding to every operation. Two dummy operations 0 and $*$ are added to mark the beginning and the end of the process. The n_1 operations of J_1 are numbered $1,\dots,n_1$ in their technological order, the n_2 operations of J_2 are numbered $n_1+1,\dots,n_1+n_2$, etc..
- $\mathcal{C}$ is the set of directed *conjunctive arcs*; they connect two operations that follow each other directly for technological reasons. Vertex 0 is connected to the set α of all first operations; the set β of all last operations is connected to vertex $*$.
- $\mathcal{D}$ is the set of *disjunctive arcs*; it contains oppositely directed arcs connecting each pair of operations in μ_ℓ, the set of operations to be performed on M_ℓ $(\ell = 1,\dots,m)$.

To each arc in both $\mathcal{C}$ and $\mathcal{D}$ a *length* p_k is assigned, corresponding to the processing time of the operation k which is its initial vertex.

A typical example of a disjunctive graph is pictured in figure 1.

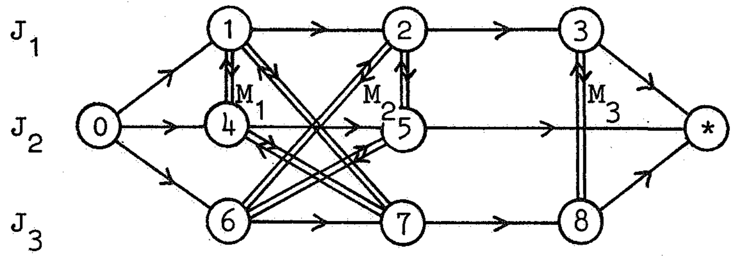

Figure 1

A disjunctive arc is *settled* if the oppositely directed arc is rejected; this means that on some machine one operation is made to precede another one.

At each stage of the algorithms we have a partial solution characterized by a subset $D \subset \mathcal{D}$ of disjunctive arcs that have been settled. We denote by $\mathcal{M}_0$ the set of machines that still have some unsettled disjunctive arcs.

We can now describe the two main types of algorithms that have been reasonably succesful in solving the job-shop scheduling problem.

A. *Charlton and Death* [2,3]

1. For each partial solution determine earliest possible starting times t_k of each operation k, disregarding all disjunctive arcs in $\mathcal{D}$ - D.
2. If either $t_j - t_k \geq p_k$ or $t_k - t_j \geq p_j$ for all pairs $(j,k) \in \mu_\ell \times \mu_\ell$ $(M_\ell \in \mathcal{M}_0)$, then this partial solution is feasible: we have a complete solution.
3. However, if both $t_j - t_k < p_k$ and $t_k - t_j < p_j$, then we have a *conflict*. We choose one of the conflicts heuristically and *branch* by settling either one or the other of the disjunctive arcs in question.
4. A *lower bound* for each of these branches is given by the longest path in the newly created directed graph, disregarding again all disjunctive arcs in $\mathcal{D}$ - D.

B. *Florian et al.* [4,5]

1. At each stage we have here a set S_0 (originally α) of operations, all of whose predecessors have been scheduled. We find an operation $k_0 \in S_0$ such that

$$t_{k_0} + p_{k_0} = \min_{k \in S_0} \{t_k + p_k\}.$$

2. If $k_0 \in \mu_{\ell'}$, we *branch* by consecutively processing first all operations $k' \in S_0 \cap \mu_{\ell'}$.
3. For each of these branches we compute a *lower bound* by means of the following steps.
   a. Determine earliest possible starting times t_k of all operations $k \in \mu_\ell$ $(M_\ell \in \mathcal{M}_0)$, disregarding all disjunctive arcs in $\mathcal{D}$ - D.
   b. Also determine *tails* q_k for each operation $k \in \mu_\ell$ $(M_\ell \in \mathcal{M}_0)$; q_k is equal to the sum of processing times of all operations that follow k.
   c. For each $M_\ell \in \mathcal{M}_0$, solve the one-machine problem where operations are available at t_k, take p_k to process and have tails q_k before they are finished; an efficient branch-and-bound algorithm is available for this purpose [5]. Denote the minimum time needed to completely finish all operations on M_ℓ by C_ℓ.

d. A lower bound is given by $\max\{C_\ell\}$.

A complete example of the latter calculation can be found in [6].

3. Comments

From the above summary, two points should be clear.

First, the branching strategy of Charlton and Death is much superior to that of Florian. In the latter's algorithm all possible conflicts are settled; many of them may never really arise.

Secondly, the lower bound of Florian is much stronger than that of Charlton and Death. This is amply confirmed by actual tests [5]. Even a much weaker version of Florian's algorithm [4] that restricted itself to those machines M_ℓ with $M_\ell \cap \beta \neq \emptyset$ and consequently ignored the tails, was superior to any other algorithm existing at that time. The algorithm as sketched in section 2 is obviously more complex from a computational point of view, but the increased strength of the lower bound makes this algorithm again superior to the former one. Thus we reach the very important conclusion that *it is worth wile to spend some extra computation time in order to find stronger bounds and reduce the search tree as much as possible*.

It seems therefore interesting to look for a branch-and-bound algorithm in which

- the branching strategy is equivalent to that of Charlton and Death: if conflicts exist in the present partial solution, branch on one of them and proceed along the branch with the lowest lower bound;
- the computation of the lower bound is equivalent to that of Florian.

4. Towards a better algorithm?

Given a partial solution where a subset $D \in \mathcal{D}$ of disjunctive arcs has been settled, we first consider the computation of a lower bound.

By Kelly's well-known critical-path algorithm we first determine for each operation k on $M_\ell \in \mathcal{M}_0$ the earliest possible starting time t_k. The length of the critical path is equal to t_*, the earliest possible starting time of vertex *. If the partial solution is feasible, t_* is also the time needed to process all the jobs.

We next have to define the tails q_k. In Florian's algorithm q_k was taken to be equal to the sum of the remaining processing times. Here, however, we can define q_k to be equal to the length of the longest path from vertex k to vertex * minus the processing time p_k of operation k. We can easily find q_k by working backwards from vertex * to find the latest possible starting time T_k of vertex k. Then q_k is given by

$$q_k = t_* - T_k - p_k \tag{1}$$

Just as in Florian's algorithm we now want to solve the one-machine problem on each machine $M_\ell \in \mathcal{M}_0$.

There is an important difference, however. Within our set-up it is quite possible that one or more disjunctive arcs on M_ℓ have been settled during a previous branching operation. Suppose for instance that the disjunctive arc from operation j to operation k has been settled. Is it now possible that an optimal solution to the one-machine problem inevitably has k preceding j?

It is easy to see that at least operation k will not be preceding operation j directly. Indeed, if the disjunctive arc from j to k has been settled, we have obviously

$$t_k \geq t_j + p_j \tag{2}$$

and

$$p_j + q_j \geq p_j + p_k + q_k$$

or

$$q_j \geq p_k + q_k \tag{3}$$

Now if k would precede j directly (figure 2(a)), we would interchange the two operations (figure 2(b)) and retain a feasible schedule because of (2). It cannot have got worse, since j now finishes earlier than previously but still not before k because of (3); all other operations have not been moved.

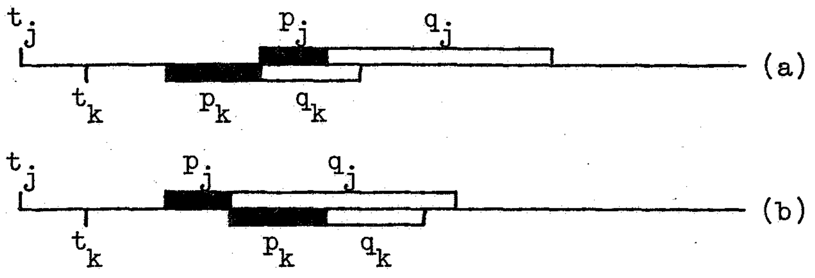

Figure 2

This result is, however, not extendable to a more general result. A counter-example is constructed in what follows.

It is easy to see that we need at least 5 operations for this example. In the optimal solution operation k has to be preceded by at least one other operation because otherwise j could be inserted before k in view of (2); likewise j has to be followed by at least one operation in view of (3), and k and j have to be separated by at least one operation because of the reasoning above.

We now construct an example with 5 operations where the only optimal solution inevitably contradicts a previously settled disjunctive arc. Suppose the disjunctive arc from 1 to 2 has been settled; the further data are given in table 1. Then the only optimal solution to this particular one-machine problem is given by the sequence (3,2,4,1,5); see figure 3.

k	t_k	p_k	q_k
1	0	2	3
2	2	1	2
3	0	2	5
4	3	2	6
5	7	2	2

Table 1

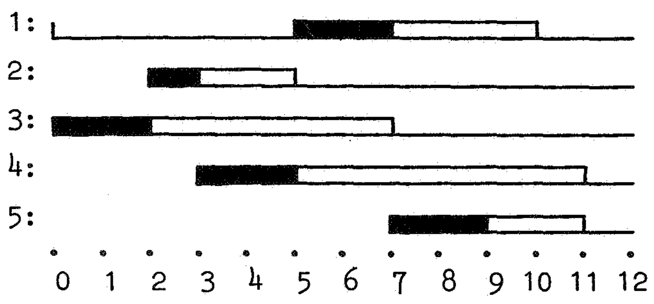

Figure 3

From the preceding discussion, it is clear that we have to take already settled disjunctive arcs into account while solving the one-machine problems. This will effectively increase the bound. No particular problems are expected in adapting the one-machine solution algorithm to these added precedence constraints. Although an extra feasibility check has to be added, the total number of feasible solutions is substantially reduced. The net effect of these two changes remains to be seen.

Having found optimal values C_ℓ for each $M_\ell \in \mathcal{M}_0$, a lower bound LB is given by LB = $\max\{C_\ell\}$.

We proceed along that branch among those created recently, that has the lowest lower bound. We now want to find out if this particular (partial) solution is feasible. We could do this by using the Charlton-and-Death criterium for a conflict which would require each operation to be able to start at the earliest possible starting time t_k. It is, however, perfectly possible that an operation k starts after t_k but that the overall schedule is still feasible in the sense that all operations can be finished before t_* and all previously settled disjunctive arcs are respected. So we settle for a broader definition of conflict that is more complex from a computational point of view, but will hopefully further reduce the search tree - something that is badly needed indeed (see the final remark in [5]).

In searching for this conflict, *we want to make as much use as possible of the optimal sequence found on each machine* M_ℓ *during the lower bound computation*. We divide the search for a possible conflict in two stages:

1. First, we look at each machine M_ℓ to find out if there is a sequence of operations on M_ℓ that allows every operation k to start on T_k at the latest. (If this is the case we say that there is a *1-feasible* solution on M_ℓ.)
2. If there is a 1-feasible solution on each M_ℓ, we try to find out if the schedules on each machine can be combined to form an overall feasible (or m-*feasible*) solution.

The following result concerns the first stage.

Theorem 1. The optimal solution to the one-machine problem on M_ℓ is 1-feasible if and only if $C_\ell \le t_*$.

Proof. Denote by B_k the starting time of operation k in the optimal solution to the one-machine problem on M_ℓ. We have by definition

$$\max_{k\in\mu_\ell} \{B_k + p_k + q_k\} = C_\ell$$

If $C_\ell \leq t_*$, then obviously

$$B_k + p_k + q_k \leq t_* \qquad \text{for all } k \in \mu_\ell \qquad (4)$$

so by (1)

$$B_k + p_k + t_* - T_k - p_k \leq t_*$$

or

$$B_k \leq T_k \qquad \text{for all } k \in \mu_\ell,$$

i.e., the solution is 1-feasible. Conversely, if $B_k \leq T_k$ for all $k \in \mu_\ell$, then (4) follows easily, and therefore $C_\ell \leq t_*$. (Q.E.D.)

Remark. We note in passing (with Florian [5]) that solving the one-machine problem with tails q_k is equivalent to solving a one-machine problem with due-dates $d_k = T_k + p_k$, where the objective is to minimize the maximum lateness L_{max}, *lateness* being defined as the difference (negative or positive) between finishing time $B_k + p_k$ and due-date d_k.

This equivalence is easily proved as follows:

$$\begin{aligned}
&\max_{k\in\mu_\ell} \{B_k + p_k + q_k\} = \\
&= \max_{k\in\mu_\ell} \{B_k + p_k + t_* - T_k - p_k\} = \\
&= t_* + \max_{k\in\mu_\ell} \{B_k + p_k - d_k\} = \\
&= t_* + L_{max}.
\end{aligned}$$

Another way of stating the above theorem is then that $C_\ell \leq t_*$ is equivalent to $L_{max} \leq 0$.

At this stage of the proceedings, there are two possibilities: either $C_\ell > t_*$ for at least one ℓ, or $C_\ell \leq t_*$ for all ℓ. We will successively consider these possibilities in what follows.

Suppose first that there are some machines, say $M_{\ell_1},\ldots,M_{\ell_s}$, for which the solution to the one-machine problem is not 1-feasible:

$$C_{\ell_j} > t_* \qquad (j = 1,\ldots,s).$$

In view of the remark above we may draw the conclusion that no 1-feasible solution on M_{ℓ_j} can then be found at all: the minimum L_{max} is strictly positive, so at least one operation will have to start after T_k.

Following the terminology of Charlton and Death, we now have a *conflict* on $M_{\ell_1},\ldots,M_{\ell_s}$. Like them, we want to branch by settling a disjunctive arc, not already in D, in either one or the other direction.

Although we are still investigating possibilities for a better branching strategy, we think the strategy described below has at least the advantage of being computationally simple and may lead to quite acceptable results.

a. Select the machine M_{ℓ_j} *for which*

$$C_{\ell_j} = \max\{C_\ell\}.$$

(This effectively reduces the number of one-machine problems that we have to solve during the lower bound computation; as soon as we have found that $C_{\ell_0} > t_*$ for some ℓ_0 we will only be interested in those machines that might conceivably produce a still higher C_ℓ.)

b. On M_{ℓ_j} *find the pair of operations* (j,k) *such that*

$$\min\{t_j + p_j - t_k, t_k + p_k - t_j\}$$

is maximal and branch by settling a disjunctive arc between j *and* k *either in one or the other direction.*

(Such a disjunctive arc cannot have been settled already, because in this case both $t_j + p_j - t_k$ and $t_k + p_k - t_j$ are non-positive.)

We realize that, in choosing this branching strategy, we do not use the information provided by the one-machine solution (except in step *a*). Though we are still exploring ways to use this information in step *b* as well, there seems at the moment no way of doing so without running into serious computational trouble.

Suppose now that $C_\ell \leq t$ for all ℓ. We would like to conclude that in this case:

$$B_k + p_k \leq B_{k+1} \tag{5}$$

for every pair of operations $(k,k+1)$ $(k \in \mu_\ell, k+1 \in \mu_{\ell'})$ *whereby* $k+1$ *directly follows* k *for technological reasons.*

If this conclusion would be justified it would immediately imply that all m 1-feasible solutions could be combined into 1 m-feasible solution.

The proof of (5) would have to be based on interfering properties of the one-machine problems. The following theorem at least assures us that the conflict between two machines will not be too serious.

Theorem 2. If operation $k \in \mu_\ell$ is directly followed by operation $k+1 \in \mu_{\ell'}$, and if $C_\ell \leq t_*$, then

$$B_k + p_k \leq T_{k+1} \tag{6}$$

Proof. Since $C_\ell \leq t_*$, we have from theorem 1:

$$t_k \leq B_k \leq T_k \tag{7}$$

Evidently (cf. (3)):

$$t_* - q_k \leq t_* - q_{k+1} - p_{k+1}$$

or

$$T_k + p_k \leq T_{k+1} \tag{8}$$

Combining (7) and (8) we get (6). (Q.E.D.)

Although the above theorem effectively bounds the seriousness of the conflict between M_ℓ and $M_{\ell'}$, such a conflict nevertheless might create nasty problems. If (5) would be correct, these problems would disappear at once. At the moment, however, we cannot present a proof of (5), nor, of course, of the equivalent statement that if $B_k + p_k > B_{k+1}$ for some $(k,k+1)$ $(k \in \mu_\ell, k+1 \in \mu_{\ell'})$ then either C_ℓ or $C_{\ell'}$ is greater than t_*. The difficulties that we encountered in trying to construct a counterexample lead us to conjecture that in most cases (5) will turn out to be true anyway. Tests of this conjecture on randomly generated examples will either lead to the desired counterexample or to an intensified search for an analytic proof.

Suppose now that (5) does not generally hold true. Theorem 2 underlines that we may then still try to set things right in the following manner. We define the *slack* S_ℓ on each machine to be equal to $t_* - C_\ell$. Now, if necessary, we can postpone all operations on M_ℓ collectively by a maximum amount of S_ℓ. (Essentially we need not postpone all operations at the same time; however, if we move one of them, we will at least also have to postpone those operations following it in the same *block*; see [5] for a definition of this term.)

The fact remains, however, that S_ℓ will often be equal to 0; to be more precise, this will certainly be the case for all machines through which a critical path is running. For each operation k on a critical path we always have $B_k + p_k + q_k \geq t_*$, so if we know that $C_\ell \leq t_*$ for all ℓ, it follows that for all these machines $C_\ell = t_*$ and $S_\ell = 0$.

Our conclusion is that we will have to check (5) for all pairs (k,k+1). If possible, we can try to set things right by using the machine slack $S_{\ell'}$; if that does not work, we will branch on two operations on M_ℓ in the way described in step *b* above.

It may be possible that by rearranging operations on some machines we get m 1-feasible solutions that can indeed be combined into 1 m-feasible one, whereas the m original solutions could not. On heuristic grounds we want to disregard this possibility for the time being.

5. Conclusions

Obviously, the better algorithm that we are looking for has not yet been completely constructed - hence the numerical index in the title of this paper. Considering the close links that we have found so far between lower bound computations and checks for feasibility, we tentatively conclude that this enquiry is worth pursuing and may indeed lead to a more forceful attack on our complicated scheduling problem.

References

1. B. ROY, B. SUSSMANN, Les problèmes d'ordonnancement avec contraints disjonctives, Note DS no.9 bis, SEMA, Paris, 1964.
2. J.M. CHARLTON, C.C. DEATH, A method of solution for general machine scheduling problems, *Operations Res.* 18(1970)689-707.
3. J.M. CHARLTON, C.C. DEATH, A generalized machine scheduling algorithm, *Operational Res. Quart.* 21(1971)127-134.
4. M. FLORIAN, P. TREPANT, G. McMAHON, An implicit enumeration algorithm for the machine sequencing problem, *Management Sci.* 17(1971)B782-792.
5. P. BRATLEY, M. FLORIAN, P. ROBILLARD, On sequencing with earliest starts and due-dates with application to computing bounds for the $(n|m|G|F_{max})$ problem, *Naval Res. Logist. Quart.* 20(1973)57-67.
6. A.H.G. RINNOOY KAN, The machine scheduling problem, Report BW 27/73, Mathematisch Centrum, Amsterdam, 1973.