REVUE FRANÇAISE D'AUTOMATIQUE, D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

FRANK WERNER

A locally optimal insertion heuristic for a one-machine scheduling problem

Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, tome 24, n° 3 (1990), p. 255-262.

<http://www.numdam.org/item?id=RO_1990__24_3_255_0>

Recherche opérationnelle/Operations Research
(vol. 24, n° 3, 1990, p. 255 à 262)

A LOCALLY OPTIMAL INSERTION HEURISTIC FOR A ONE-MACHINE SCHEDULING PROBLEM (*)

by Frank WERNER (1)

Abstract. – *In this paper we consider a one-machine scheduling problem with release dates and the minimization of the weighted number of late jobs. This problem is NP-hard. We develop a one-pass algorithm (insertion heuristic) which yields a locally optimal solution in a special left shift neighbourhood graph. Finally we investigate in which of the known polynomially solvable special cases our heuristic yields an optimal solution.*

Keywords : Combinatorial optimization; Scheduling; One-machine Problems; Heuristics.

Résumé. – *Nous considérons un problème d'ordonnancement pour une machine avec dates de mise à disposition et minimisation du nombre pondéré des travaux en retard. Ce problème est NP-dur. Nous développons un algorithme à une passe (insertion heuristique) qui donne une solution localement optimale dans un graphe spécial de voisinage par translation à gauche. Finalement, nous examinons quels sont, parmi les cas spéciaux connus comme polynomialement résolubles, ceux pour lesquels notre heuristique donne une solution optimale.*

1. INTRODUCTION

In this paper we consider a special one-machine scheduling problem: n jobs $J_i (i=1, \ldots, n)$ have to be processed on a machine. A release date r_i, a processing time t_i, a due date d_i and a weight w_i are connected with job J_i. Each solution can be described by a permutation of the job indices. If a permutation p is chosen we denote by $S_i(p)$ the starting time of J_i and by $C_i(p) = S_i(p) + t_i$ the completion time of J_i. Now we can introduce a value U_i with

$$U_i = \begin{cases} 1 & \text{if } C_i(p) > d_i \\ 0 & \text{otherwise} \end{cases}$$

(*) Received in February 1990.
(1) Technische Universität « Otto von Guericke » Magdeburg, Sektion Mathematik WOF, Postschließfach 124, Magdeburg, 3010 R.D.A.

Recherche opérationnelle/Operations Research, 0399-0559/90/03 255 08/$ 2.80

which indicates whether J_i is late or not. We look for a permutation which minimizes the objective $\sum_{i=1}^{n} w_i U_i$. This problem is denoted as $n/1/r_i \geqq 0/\Sigma w_i U_i$.

The mentioned problem is NP-hard. Therefore, enumeration methods have been developed for this problem. In the special case of $r_i = 0$ for all jobs, Potts and van Wassenhove [8] developed a branch and bound algorithm. Further exact algorithms are summarized for this case in [2]. If we have release dates $r_i \geqq 0$ and $w_i = 1$ for all jobs, the problem is already NP-hard. A dynamic programming algorithm is given by Vlach [11].

However, the development of heuristics for NP-hard problems is also going on. As regards permutation problems, insertion techniques are often applied. For instance, insertion methods for the traveling salesman problem can be found in [1] or [9], a variant for the permutation flow shop problem is contained in [7]. In Section 2 we give such an insertion algorithm for the mentioned scheduling problem and we show that the obtained solution is locally optimal in a special neighbourhood graph. This is remarkable because the complexity of finding local optima remains an open problem in general (*cf.* [5]). Further investigations about local optimality of construction methods can be found in [13]. As regards the worst case behaviour, we present an example that our heuristic solution can be arbitrarily bad. In Section 3 polynomially solvable cases are summarized and we investigate in which cases the stated heuristic yields an optimal solution.

2. THE HEURISTIC

Insertion methods start with a single job and complete step by step a subsequence by inserting a further job in the existing sequence. In this connection two questions are of interest:

(1) Which job will be inserted next into the subsequence?

(2) On which position will the chosen job be inserted?

In our algorithm we insert the jobs according to non-increasing weights. The position is determined in such a way that the maximum lateness $L_{\max} = \max\{L_i\} = \max\{C_i - d_i\}$ of all inserted jobs is minimized. Then our

insertion algorithm is as follows:

Algorithm I:

S1: Determine $i \in H := \{1, \ldots, n\}$ with maximal weight w_i (if there exist several jobs select that i with minimal d_i, if still more than one job exists choose i with minimal t_i) ; $h := 1$; $K := \varnothing$; $Z := \varnothing$; $H := H \setminus \{i\}$;

If $r_i + t_i > d_i \Rightarrow Z := Z \cup \{i\}$, $p := \varnothing$ and go to S2;

$h := h+1$; $K := K \cup \{i\}$; $p := (i)$.

S2: Determine $i \in H$ with maximal w_i (if several i exist perform as in S1).

Determine the minimal position g of an element in p such that the completion time of the corresponding job is greater than r_i ($g := h$ if such an element does not exist).

Determine $L_{\max}(p^j)$ for $j = g, \ldots, h$ where p^j is obtained by inserting i on position j in p.

Determine $u \in \{g, \ldots, h\}$ with minimal $L_{\max}(p^u)$(if u is not uniquely determined choose the largest u).

If $L_{\max}(p^u) > 0 \Rightarrow Z := Z \cup \{i\}$ and go to S3;

$h := h+1$; $K := K \cup \{i\}$; $p := p^u$;

S3 : $H := H \setminus \{i\}$; if $H \neq \varnothing \Rightarrow$ go to S2;

Let p^Z be an arbitrary permutation of the elements of Z; $p^I := (p, p^Z)$ is the heuristic solution with $F(p^I) = \Sigma \{w_j \mid j \in Z\}$.

K and Z represent the sets of indices of early and late jobs. We only mention that in the case $g > 1$ in S2 it is not necessary to consider the insertion of i on a smaller position than g since the objective value is not lower than for p^g. The complexity of the above algorithm is $O(n^2)$. The following example serves as illustration of algorithm I.

Example 1: Let

i	1	2	3	4	5	6	7
w_i	10	8	7	5	3	2	1
r_i	15	8	6	18	3	5	10
t_i	4	8	7	8	6	5	7
d_i	33	35	25	42	18	20	28

The jobs are already numbered such that they are inserted in the above order. We obtain as heuristic solution $p^I=(5, 6, 3, 2, 1, 4, 7)$ with $Z=\{7\}$ and $F(p^I)=1$. It is noted that p^I is also the optimal solution.

As regards a worst case analysis, the following example shows that the quotient $F(p^I)/F(p^{opt})$ can be arbitrarily large where p^{opt} denotes an optimal permutation.

Example 2: Let

i	1	2	3	...	$n-1$	n
w_i	2	1	1	...	1	1
r_i	0	0	0	...	0	0
t_i	n	1	1	...	1	1
d_i	n	1	2	...	$n-2$	$n-1$

Obviously we have the optimal solution $p^{opt}=(2, 3, \ldots, n-1, n, 1)$ with $F(p^{opt})=2$. Algorithm I yields $p^I=(1, 2, \ldots, n-1, n)$ with $F(p^I)=n-1$. Hence $F(p^I)/F(p^{opt})=(n-1)/2$ unbounded increases with the number of jobs.

Now we prove that the heuristic solution is locally optimal in a special neighbourhood graph. We consider the directed graph $G(\mathrm{LS}, n)$ of left shift transformations. A neighbour p' of $p=(p_1, \ldots, p_n)$ (this means we have an arc from p to p') is obtained by shifting an arbitrary element to the left, *i.e.* this element is reinserted on a smaller position in p. Some properties of this graph are derived in [12] or [13]. Now we can formulate the following theorem.

THEOREM 1: *The heuristic solution $p^I=(p_1, \ldots, p_n)$ obtained by algorithm I is locally optimal in $G(LS, n)$.*

Proof: Assume p^I is not locally optimal, *i.e.* there exists a neighbour p' in $G(LS, n)$ with $F(p')<F(p^I)$ where the i-th element of p^I is shifted to the left. Let K and Z be the sets of job indices of early and late jobs according to p^I (*cf.* algorithm *I*). Moreover let U be the set of all elements which have been considered for insertion before p_i, $U^*=(U\cap K)\cup\{p_i\}$ as well as $\bar{p}$ the obtained permutation if in p' all elements have been deleted which do not belong to U^*.

Obviously, $L_{\max}(\bar{p})>0$ holds since $\bar{p}$ was also considered when inserting p_i but p_i was taken into Z. On the other hand we have $C_h(p')\geqq C_h(\bar{p})$ for all $h\in U^*$. Hence $\max\{L_h(p')\mid h\in U^*\}>0$ is obtained. Consequently, there exists a $u\in U^*$ with $C_u(p')>d_u$. Considering $w_u\geqq w_i$ we obtain $F(p')\geqq F(p^I)$ which contradicts the assumption. ■

3. POLYNOMIALLY SOLVABLE CASES AND THE HEURISTIC

In this section we summarize polynomially solvable cases and we investigate in which cases our heuristic yields an optimal solution. In the following it is assumed that there exists a numbering of the jobs that one of the following conditions is satisfied:

(1) $n/1/r_i \geqq 0/\Sigma w_i U_i$

(*a*) $r_1 \leqq r_2 \leqq \ldots \leqq r_n$; $d_1 \leqq d_2 \leqq \ldots \leqq d_n$;
$t_1 \leqq t_2 \leqq \ldots \leqq t_n$; $w_1 \geqq w_2 \geqq \ldots \geqq w_n$;

(*b*) $r_1 \leqq r_2 \leqq \ldots \leqq r_n$; $d_1 \leqq d_2 \leqq \ldots \leqq d_n$; $w_1 \geqq w_2 \geqq \ldots \geqq w_n$;
for all $i=1, \ldots, n-1$ holds: $t_i \leqq r_{i+1} - r_i$;

(*c*) $d_1 \leqq d_2 \leqq \ldots \leqq d_n$; $w_1 \leqq w_2 \leqq \ldots \leqq w_n$; $r_i = (i-1) \cdot t$, $i=1, \ldots, n$, $t>0$;
for all $i=1, \ldots, n$ holds: $2 \cdot (n-i) \cdot t \leqq t_i \leqq 2 \cdot (n-i) \cdot t + t$;

(2) $n/1/r_i \geqq 0/\Sigma U_i$

$r_1 \leqq r_2 \leqq \ldots \leqq r_n$; $d_1 \leqq d_2 \leqq \ldots \leqq d_n$;

(3) $n/1//\Sigma w_i U_i$

$t_1 \leqq t_2 \leqq \ldots \leqq t_n$; $w_1 \geqq w_2 \geqq \ldots \geqq w_n$;

(4) $n/1//\Sigma U_i$.

Case 4 was solved by Moore [6]. Lawler [4] stated an $O(n \log n)$ algorithm for case 3. Kise *et al.* [3] solved case 2 by an $O(n^2)$ algorithm. The remaining cases were solved by Tanaev *et al.* [10]. The following theorem by Tanaev *et al.* allows a unique consideration of all mentioned special cases.

THEOREM 2: *There exists a numbering of the jobs such that* $r_1 \leqq r_2 \leqq \ldots \leqq r_n$ *and* $d_1 \leqq d_2 \leqq \ldots \leqq d_n$. *Moreover, let* $p=(1, 2, \ldots, k)$ *with* $C_h(p) \leqq d_h$ *for* $1 \leqq h \leqq k-1$ *and* $C_k(p) > d_k$. *If there exists an* r *with* $1 \leqq r \leqq k$ *such that*

$$C_u(p \setminus r) \leqq C_v(p \setminus s) \qquad \text{and} \qquad w_r \leqq w_s \tag{1}$$

where $p \setminus r$ *denotes that permutation obtained by deleting* r *in* p *and* u, v *are the last elements in* $p \setminus r$ *and* $p \setminus s$ $(1 \leqq s \leqq k)$, *then there exists an optimal permutation in which job* J_r *is late.*

In all cases a job can be stated which statisfies (1). After selecting this job, we apply the above theorem to the remaining job set (we denote this procedure by algorithm T). Therefore, in cases 1 *a* and 1 *b* we select each time the last job and in case 1 *c* the first job. As regards the cases 3 and 4, the job with the largest processing time is selected according to Theorem 2. In case 2 the selected job is determined by comparison of all possibilities.

We now show that our insertion method yields an optimal solution in the cases 1 *a*-1 *c* whereas in cases 2-4 an optimum is not necessarily obtained.

THEOREM 3: *Algorithm I generates an optimal solution in the cases* 1 *a and* 1 *b.*

Proof: In both cases, algorithms T and I insert the jobs in the same order. Firstly, we put the first job in both algorithms either into K or into Z (denotions according to algorithm I). We assume that both algorithms have $i-1$ jobs inserted in the same manner, and $p=(p_1, \ldots, p_k)$ is the permutation of the elements of K obtained by both algorithms. We consider the insertion of i according to I. Let p^j be the obtained permutation if element i is inserted on position j in p. It is sufficient to prove that in the case $k \geqq 1$

$$L_{\max}(p^{k+1}) \leqq L_{\max}(p^j) \text{ holds for } j=1, \ldots, k.$$

Case 1 *a* : $p^j (1 \leqq j \leqq k)$ and p^{k+1} have the following form:

p^j	A	i	B	$u:=p_k$
p^{k+1}	A	B	u	i

where $A=\{p_1, \ldots, p_{j-1}\}$, $B=\{p_j, \ldots, p_{k-1}\}$ and $D=B \cup \{i, u\}$. If we have no idle time within the processing of the jobs of the index set D according to p^{k+1} we obtain because of $r_i \geqq r_v$ for $v \in B$

$$S_i(p^j) \geqq \min\{S_v(p^{k+1}) \mid v \in B\}$$

and consequently $C_u(p^j) \geqq C_i(p^{k+1})$.

In the other case let g be the element of D on the largest position in p^{k+1} such that we have an idle time before J_g. Then $S_g(p^{k+1})=r_g \leqq r_i=S_i(p^j)$ holds and hence we have $C_u(p^j) \geqq C_i(p^{k+1})$.

Furthermore we obtain

$$L_u(p^j)=C_u(p^j)-d_u \geqq C_u(p^j)-d_i \geqq C_i(p^{k+1})-d_i=L_i(p^{k+1}).$$

Moreover we have $L_v(p^j) \geqq L_v(p^{k+1})$ for all v except i. Hence both algorithms T and I add element i to p if $L_{\max}(p^{k+1})<0$ or put it into Z otherwise.

Case 1 *b*: In this case $C_u(p^j) \geqq C_i(p^{k+1})$ also holds. Therefore the proof is quite similar. ■

THEOREM 4: *In case* 1 *c algorithm I yields an optimal solution.*

Proof: Let p^T be the optimal solution obtained by algorithm T. We only have to consider the case that in p^T $k \geqq 1$ jobs meet their due dates. Then p^T has the form $p^T=(n-k+1, \ldots, n, p^*)$ where p^* is an arbitrary permutation

of the integers 1, ..., $n-k$. Algorithm I inserts the jobs in the order J_n, J_{n-1}, ..., J_1. Obviously n is taken into K by I. We assume that algorithm I has generated the permutation $p=(i+1, \ldots, n)$ and we prove that in the case $i \geqq n-k+1$ the element i is placed on position 1 by I. Let p^j be that permutation where i is inserted on position j in $p\,(1 \leqq j \leqq n-i+1)$. It is sufficient to show that for $j \geqq 2$ $L_{\max}(p^j) > L_{\max}(p^1)$ holds. p^1 and p^j have the following form:

p^1	i	A	B
p^j	A	i	B

where $A=\{i+1, \ldots, i+j-1\}$ and $B=\{i+j, \ldots, n\}$. Because of the conditions of case 1 c we have no idle time in p^1. From this and $r_i < \min\{r_h \mid h \in A\}$ follows $C_h(p^1) < C_i(p^j)$ for all $h \in A$. Considering $d_i \leqq d_h$ for $h \in A$ we obtain $L_i(p^j) = C_i(p^j) - d_i > C_h(p^1) - d_i \geqq C_h(p^1) - d_h = L_h(p^1)$. Furthermore, we have $L_i(p^j) > L_i(p^1)$ and in the case $B \neq \varnothing$ $L_h(p^j) > L_h(p^1)$ for all $h \in B$. Hence $L_{\max}(p^j) > L_{\max}(p^1)$ holds. Therefore we insert i on position 1 in p since J_i is not late in p^T and this means $L_{\max}(p^1) \leqq 0$.

The first $n-k$ jobs are taken into Z by I because p^T is an optimal solution. ■

Example 3: Let

i	1	2	3
w_i	1	1	1
r_i	0	0	0.
t_i	25	10	15
d_i	30	32	35

Algorithm I yields $p=(1, 2, 3)$ with $F(p)=2$ and $Z=\{2, 3\}$. On the other hand algorithm T generates the optimal solution $p^{opt}=(2, 3, 1)$ with $F(p^{opt})=1$ and $Z=\{1\}$. Hence by algorithm I an optimal solution is not necessarily obtained in the cases 2-4.

4. CONCLUDING REMARKS

We stated an insertion heuristic for a special one-machine scheduling problem. For instance, such fast heuristics can be applied in enumeration methods for determining upper bounds. It was proved that the obtained solution is locally optimal in a left shift neighbourhood graph. The performed

structural analysis is possibly an alternative to the often pessimistic results of the worst case analysis for one-machine problems. Moreover, from such structural analysis we obtain recommendations which kind of iteration method can be added to a specific one-pass algorithm. It also could be a topic of further investigations whether local optimality can be used for deriving other enumeration strategies.

REFERENCES

1. B. GOLDEN, L. BODIN, T. DOYLE and W. Jr. STEWART, Approximate Traveling Salesman Algorithms, *Oper. Res.*, 1980, *28,* pp. 694-711.
2. S. K. GUPTA and J. KYPARISIS, Single Machine Scheduling Research, *OMEGA*, 1987, *15,* pp. 207-227.
3. H. KISE, T. IBARAKI, and H. MINE, A Solvable Case of the One-Machine Scheduling Problem with Ready and Due Times, *Oper. Res.*, 1978, *26,* pp. 121-126.
4. E. L. LAWLER, Sequencing to Minimize the Weighted Number of Tardy Jobs, *R.A.I.R.O. Rech. Oper.* 1976, *10,* pp. 27-33.
5. D. C. LLEWELLYN, C. TOVEY and M. TRICK, Local Optimization on Graphs, *Discrete Appl. Math.*, 1989, *23,* pp. 157-178.
6. J. M. MOORE, An *n* Job, one Machine Sequencing Algorithm for Minimizing the Number of Late Jobs, *Manage. Sci.*, 1968, *15,* pp. 102-109.
7. M. NAWAZ, E. E. ENSCORE and I. HAM, A Heuristic Algorithm for the *m*-Machine, *n*-Machine Flow-Shop Sequencing Problem, *OMEGA*, 1983, *11,* pp. 91-95.
8. C. N. POTTS and L. N. WASSENHOVE, Algorithms for Scheduling a Single Machine to Minimize the Weighted Number of Late Jobs, *Manage. Sci.*, 1988, *34,* pp. 843-858.
9. D. J. ROSENKRANTZ, R. E. STEARNS and P. M. LEWIS, An Analysis of Several Heuristics for the Traveling Salesman Problem, *SIAM J. Comput.*, 1977, *6,* pp. 563-581.
10. V. S. TANAEV, V. S. GORDON and J. M. SHAFRANSKI, Theory of Scheduling: One-Processing Systems (Russian), Moscow, 1984.
11. M. VLACH, One-Processor Scheduling with Non-Uniform Ready Times, *M.O.S. Ser. Optimization*, 1979, *10,* pp. 93-96.
12. F. WERNER, Zu einigen Nachbarschaftsgraphen für die Entwicklung geeigneter Iterationsverfahren zur näherungsweisen Lösung eines speziellen Permutationsproblems, *Wiss. Zeitschrift TU Magdeburg*, 1987, H. 4, *31,* pp. 48-54.
13. F. WERNER, Zur Struktur und näherungsweisen Lösung ausgewählter kombinatorischer Optimierungsprobleme, *Dissertation B*, TU Magdeburg, 1989.