Operations Research Letters 18 (1996) 167–170

List scheduling revisited

J.M.J. Schutten [*]

Faculty of Mechanical Engineering, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Received 1 August 1994; revised 1 October 1995

Abstract

We consider the problem of scheduling n jobs on m identical parallel machines to minimize a regular cost function. The standard list scheduling algorithm converts a list into a feasible schedule by focusing on the job start times. We prove that list schedules are dominant for this type of problem. Furthermore, we prove that an alternative list scheduling algorithm, focusing on the completion times rather than the start times, yields also dominant list schedules for problems with sequence dependent setup times.

Keywords: Scheduling; List scheduling; Parallel machines; Setup times; Regular cost functions

1. Introduction

We consider the problem of scheduling a set $\mathcal{J}$ of n independent jobs $J_1, J_2, \ldots, J_n$ on m identical parallel machines $M_1, M_2, \ldots, M_m$ with *sequence dependent* setup times. Each job J_j ($j = 1, \ldots, n$) must be processed without preemption on exactly one of the machines during a given non-negative time p_j and may have a release date and a due date. Each machine M_k ($k = 1, \ldots, m$) is available from a given non-negative time S_k onwards and can process at most one job at a time. A *schedule* σ specifies for each job J_j a completion time $C_j(\sigma)$. The quality of schedule σ is measured by a *regular* objective function $f(\sigma)$ that needs to be minimized. An objective function f is called regular if $f(\sigma_1) > f(\sigma_2)$ implies that $C_j(\sigma_1) > C_j(\sigma_2)$ for at least one j ($j = 1, \ldots, n$); cf. Baker [1]. For any scheduling problem with a regular objective function, there always exists an optimal schedule that is *semiactive*. A semiactive schedule is a schedule in which no job can be processed earlier without changing the job sequences on the machines.

In the remainder, we use the three-field notation proposed by Graham et al. [5] to classify machine scheduling problems. The first field in this notation specifies the machine environment. The symbols we use are P and Q to indicate identical and uniform parallel machines, respectively. The second field specifies the job characteristics. The symbols we use are r_j to indicate that jobs have release dates, and s_i and s_{ij} to indicate that sequence independent setup times and sequence dependent setup times, respectively, occur. The last field specifies the objective function. The symbols we use are $C_{\max}$ to indicate the minimization of the makespan, and $L_{\max}$ to indicate the minimization of the maximum lateness.

The plan of this paper is as follows. In Section 2, we explain the concept of list scheduling. We point out that list schedules need not be dominant for problems with sequence dependent setup times if the list

[*] E-mail: J.M.J. Schutten @ wb.utwente.nl.

Table 1
Processing times

J_j	p_j
J_1	1
J_2	2
J_3	3

Table 2
Setup matrix s_{ij}

j	i		
	1	2	3
0	1	1	10
1	0	10	10
2	10	0	1
3	10	10	0

scheduling algorithm focuses on the starting times of the jobs. Finally, in Section 3, we prove that the list schedules are dominant if the list scheduling algorithm focuses on the completion times of the jobs.

2. Standard list scheduling

Given a certain list or permutation π of the job set $\mathcal{J}$, a standard list scheduling algorithm constructs a schedule for the parallel machines as follows: the next job of the list is scheduled on the machine that is available first. If a tie exists, then usually the job is scheduled on the machine with the smallest index. Thus, this algorithm focuses on the starting times of the jobs. The schedule that results from the list π is denoted by LIST(π). Several authors analyze the worst-case performance of list scheduling algorithms. For instance, Graham [4] analyzes the worst-case performance of the list scheduling algorithm with the jobs sorted in order of non-increasing processing times for the problem $P||C_{\max}$. List schedules are also used in branch-and-bound algorithms for problems in which the set of list schedules is *dominant*, i.e., contains at least one optimal solution. Woerlee [8], for instance, develops such a branch-and-bound algorithm for the problem $P|r_j|L_{\max}$.

A schedule in which no machine is kept idle when there is a job available for processing is called a *non-delay* schedule. Given a non-delay schedule σ, there is exactly one list π such that LIST(π)$=\sigma$: the ith job of π is the job with the ith smallest starting time in σ. So, there is a one-to-one relation between a non-delay schedule and a list. Therefore, if for a certain problem the non-delay schedules are dominant, then enumerating all possible lists and evaluating the resulting list schedules yields an optimal solution; see also Elmaghraby and Park [3].

For problems with setup times, however, the non-delay schedules are not dominant. Ovacik and Uzsoy [6] present an instance for the problem $P|s_{ij}|C_{\max}$ for which the set of list schedules contains no optimal solution. The data for this instance with two machines and three jobs are given in Tables 1 and 2. The optimal solution with makespan 7 is given in Fig. 1. In no optimal schedule is J_3 processed first. So, the optimal list must be $J_1-J_2-J_3$ or $J_2-J_1-J_3$. Neither list yields, however, an optimal solution. For problems with general release and due dates the non-delay schedules are not dominant either.

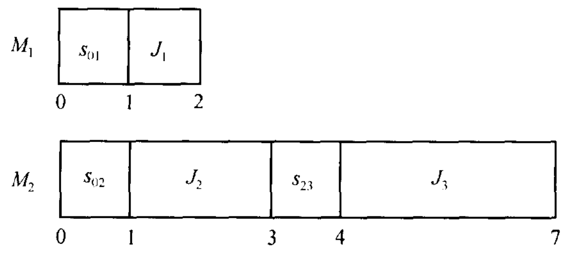

Fig. 1. Gantt chart representing the optimal solution.

In the following section, we present an alternative list scheduling algorithm. This algorithm focuses on the completion times of the jobs instead of the starting times. For problems without setup times, this alternative list scheduling algorithm is equal to the standard list scheduling algorithm. We show that the list schedules are dominant, even for problems with sequence dependent setup times.

3. An alternative list scheduling algorithm

Recall that standard list scheduling algorithms assign the next job of the list to the machine that is *available* first. In the alternative list scheduling algorithm we assign the next job of the list to the machine on which it is *completed* first. The schedule that results from list π using the latter algorithm is denoted by $g(\pi)$. Note that for problems without setup times $\mathrm{LIST}(\pi) = g(\pi)$ for every list π. Note also that $g(\pi)$ is computed in $O(nm)$ time, whereas $\mathrm{LIST}(\pi)$ takes $O(n \log m)$ time. Cho and Sahni [2] also use a list scheduling algorithm that focuses on the completion times. They give a worst-case performance of this algorithm for the problem $Q||C_{\max}$ and special cases of it.

The next theorem proves that the list schedules obtained with the alternative list scheduling algorithm are dominant for a broad class of parallel machine problems.

Theorem 1. *Suppose a set of jobs needs to be scheduled on identical parallel machines. The jobs have release dates, setup times need to be taken into account, and some regular cost function needs to be minimized. Then, there exists a list π such that $g(\pi)$ is an optimal schedule.*

Before we prove this theorem, we introduce some additional notation. Let π be any list of a subset of $\mathcal{J}$. $g(\pi)$ is then a *partial* schedule. Denote by $n_i(\sigma)$ $(i = 1,\dots,m)$ the number of jobs that are scheduled on machine M_i in the (partial) schedule σ and let Ω be the set of optimal complete schedules. We say that a partial schedule σ' *deviates* from a complete schedule σ if one of the following conditions holds:

(1) $n_i(\sigma') > n_i(\sigma)$ for at least one i $(1 \leqslant i \leqslant m)$;

(2) the jth job on machine M_i in σ' is not the jth job on machine M_i in σ for some i and j $(1 \leqslant i \leqslant m;\ 1 \leqslant j \leqslant n_i(\sigma'))$.

Let $\tau(\sigma)$ be the maximum number of jobs that any list π can contain such that $g(\pi)$ does not deviate from a given σ, that is,

$$\tau(\sigma) = \max_{\pi \in \Pi}\{|\pi| \mid g(\pi) \text{ does not deviate from } \sigma\},$$

where Π is the set of all possible lists of subsets of $\mathcal{J}$ and $|\pi|$ is the number of jobs in π. We are now ready to prove the theorem. We do this by contradiction.

Proof of Theorem 1. Let $\sigma^* \in \Omega$ be any optimal schedule for which

$$\tau(\sigma^*) = \max_{\sigma \in \Omega} \tau(\sigma).$$

If the theorem does not hold, then $\tau(\sigma^*) < n$. Let π^* be such that it contains $\tau(\sigma^*)$ jobs and $g(\pi^*)$ does not deviate from σ^*.

Consider the directed graph $D = (V, A)$ with a node $v_i \in V$ for each machine M_i $(i = 1,\dots,m)$. Suppose that $n_i(\sigma^*) > n_i(g(\pi^*))$. Let $J_{[i]}$ be the first job on M_i in σ^* that is not in $g(\pi^*)$ and let R_i be the job sequence that consists of $J_{[i]}$ and its successors on M_i in σ^*. Note that R_i consists of precisely those jobs on M_i in σ^* that are not in $g(\pi^*)$. Suppose that $J_{[i]}$ is scheduled on machine M_j in $g(\pi^* J_{[i]})$. Then we have that $j \neq i$, because $\tau(\sigma^*)$ is maximal. Draw an arc in D from v_i to v_j. Do this for every machine M_i with $n_i(\sigma^*) > n_i(g(\pi^*))$. We can distinguish two cases:

(1) There is a $v_j \in V$ with an incoming arc and no outgoing arc. Say, $(v_i, v_j) \in A$. Then we know that in σ^* and in $g(\pi^*)$ exactly the same jobs are scheduled on M_j, because otherwise v_j would have had an outgoing arc. Also,

$$C_{[i]}(g(\pi^* J_{[i]})) \leqslant C_{[i]}(\sigma^*)$$

due to the way $J_{[i]}$ is assigned to a machine in $g(\pi^* J_{[i]})$. Let σ' be the schedule obtained from σ by moving the sequence R_i to M_j. σ' is also optimal, because $C_{[i]}(\sigma') \leqslant C_{[i]}(\sigma)$, and therefore $C_j(\sigma') \leqslant C_j(\sigma)$ for $j = 1,\dots,n$. What is more, $g(\pi^* J_{[i]})$ does not deviate from $\sigma' \in \Omega$, which is a contradiction with the maximality of $\tau(\sigma^*)$.

(2) There is no $v_j \in V$ with an incoming arc and no outgoing arc. This means that each $v_j \in V$ has either an outgoing arc, or neither an incoming nor an outgoing arc. Then D must contain at least one directed cycle K. Without loss of generality, we assume that $K = v_1 v_2 \dots v_p v_1$. Let σ' be the schedule obtained from σ by moving the sequences $R_1,\dots,R_p$ as follows: move sequence R_1 to machine M_2, R_2 to $M_3,\dots,R_{p-1}$ to M_p, and R_p to M_1. Using the same arguments as in case 1, we conclude that σ' is optimal, too. Since $\pi^* J_{[1]}$ does not deviate from σ', this is, again, a contradiction with the maximality of $\tau(\sigma^*)$. □

Theorem 1 implies that for many parallel machine problems a set of at most $n!$ schedules is dominant. Note that different lists may result in the same schedule. Dominance rules that prevent exploring several lists that result in the same schedule can even further reduce this set. A branch-and-bound algorithm that uses the alternative list scheduling algorithm and such dominance rules has been successfully developed by Schutten and Leussink [7] for the problem $P|r_j, s_i|L_{\max}$.

The proof depends on the condition that $C_{[i]}(\sigma') \leqslant C_{[i]}(\sigma^*)$ implies that $C_j(\sigma') \leqslant C_j(\sigma^*)$ for all jobs J_j in the sequence R_i. This condition does not hold for non-identical parallel machine scheduling problems. Hence, the theorem does not hold for uniform and unrelated parallel machine scheduling problems.

References

[1] K.R. Baker, *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

[2] Y. Cho and S. Sahni, "Bounds for list schedules on uniform processors", *SIAM J. Comput.* **9**, 91–103 (1980).

[3] S.E. Elmaghraby and S.H. Park, "Scheduling jobs on a number of identical machines", *AIIE Trans.* **6**, 1–13 (1974).

[4] R.L. Graham, "Bounds on multiprocessor timing anomalies", *SIAM J. Appl. Math.* **17**, 416–429 (1969).

[5] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Ann. Discrete Math.* **5**, 287–326 (1979).

[6] I.M. Ovacik and R. Uzsoy, "Worst-case error bounds for parallel machine scheduling problems with bounded sequence-dependent setup times", *Oper. Res. Lett.* **14**, 251–256 (1993).

[7] J.M.J. Schutten and R.A.M. Leussink, "Parallel machine scheduling with release dates, due dates and family setup times", *Internat. J. Prod. Econom.* (1996) forthcoming.

[8] A.P. Woerlee, "Decision Support Systems for Production Scheduling", Ph.D. Thesis, Erasmus University, Rotterdam, 1991.