

Local search for multiprocessor scheduling: how many moves does it take to a local optimum?

C.A.J. Hurkens * T. Vredeveld †

August 30, 2002

Abstract

We analyze two local search algorithms for multiprocessor scheduling. The first algorithm is a job interchange algorithm for identical parallel machines due to Finn and Horowitz [Bit 19; 1979]. We construct instances for which this algorithm takes a quadratic number of iterations. This contradicts the original analysis of Finn and Horowitz who claimed a linear number of iterations.

The second algorithm adds an additional rule to the Finn and Horowitz algorithm. Even for n jobs on m uniformly related machines, this modified algorithm takes only $O(nm)$ iterations.

1 Introduction

In this note we consider multiprocessor scheduling problems. We are given a set of n jobs, $1, \dots, n$, with positive integer processing requirements p_j ($j = 1, \dots, n$). Each of these jobs has to be processed without interruption on one of the m given machines, M_1, \dots, M_m . A machine can process at most one job at a time and all machines and jobs are available from the beginning. The objective is to minimize the makespan, i.e., we want the last job to complete as early as possible. We consider two machine environments: *identical* and *uniform* parallel machines. In the case of identical machines, the time it takes to fully process a job j on any machine is equal to the jobs processing requirement. In the uniform parallel machine environment, each machine has a given positive integral speed s_i ($i = 1, \dots, m$). The time needed to complete a job j on a machine M_i is equal to p_j/s_i . The *load of a machine* is the total processing time assigned to it. A *critical machine* is a machine with maximum load and the makespan is equal to the maximum machine load. These problems are well known to be NP-hard, even in the case for two identical machines, see Lenstra, Rinnooy Kan, and Brucker (1977).

*wscor@win.tue.nl. Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands.

†tjark@win.tue.nl. Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands; supported by the project “High performance methods for mathematical optimization” of the Netherlands Organization for Scientific Research (NWO)

procedure 0/1-INTERCHANGE
initialize S by placing the n jobs on the m machines in any order
while there is a job $j \in J_{\max}$ with $p_j < L_{\max} - L_{\min}$
 modify S by moving job j to a machine M_i with $L_i = L_{\min}$
end while

Figure 1: The Finn-Horowitz 0/1-INTERCHANGE algorithm

A schedule is denoted by $S = (J_1, \dots, J_m)$, where J_i denotes the set of jobs scheduled on machine M_i ($i = 1, \dots, m$). J_{\max} denotes the set of jobs scheduled on the critical machines. The load of a machine is $L_i = \sum_{j \in J_i} p_{ij}$, where $p_{ij} = p_j$ for the problem on identical parallel machines and $p_{ij} = p_j/s_i$ for the problem on uniform parallel machines. As the makespan is equal to the maximum machine load, we denote it by $L_{\max} = \max_{1 \leq i \leq m} L_i$ and the minimum machine load is $L_{\min} = \min_{1 \leq i \leq m} L_i$.

Finn and Horowitz (1979) presented for the problem on identical parallel machines a simple improvement heuristic, which they called 0/1-INTERCHANGE. For this procedure, we define a job j as a *qualifying job* if it is scheduled on a critical machine and has $p_j < L_{\max} - L_{\min}$. The procedure iteratively moves a qualifying job to a machine with minimal load, until there are no qualifying jobs; see Figure 1 for a more formal description. Note that the algorithm needs a *job selection rule* for selecting which qualifying job to move. 0/1-INTERCHANGE finds a so-called *jump-optimal* solution, i.e., a solution that cannot be improved by moving one job to another machine. Finn and Horowitz showed that the ratio of the makespan of the obtained schedule to the optimum is never worse than $\frac{rm}{rm-m+1}$, where r denotes the number of jobs on a critical machine. If there is a critical machine processing only one job, the schedule is optimal. For $r \geq 2$, the ratio is bounded from above by $\frac{2m}{m+1}$, see also Schuurman and Vredeveld (2001). Finn and Horowitz claimed that, regardless of the job selection rule, the 0/1-INTERCHANGE procedure terminates in $\mathcal{O}(n)$ iterations. In Section 2, we contradict this claim by providing an example and job selection rule for which 0/1-INTERCHANGE needs $\Omega(n^2)$ iterations. Brucker, Hurink, and Werner (1997) showed that the procedure terminates in $\mathcal{O}(n^2)$ iterations, regardless of the job selection rule.

Finn and Horowitz also proposed an implementation of their algorithm. For each machine, the jobs scheduled on the machine are stored in a linked list. The procedure scans the list of a critical machine, starting at the beginning, and moves the first encountered qualifying job. This job is added to the rear of the list of a machine with minimum load. When a non-qualifying job is scanned, we know that this job cannot become a qualifying job in future iterations as $L_{\max} - L_{\min}$ is non-increasing. We adapt the previously mentioned example so that the proposed implementation of 0/1-INTERCHANGE needs $\Omega(n^2)$ iterations.

In Section 3, we consider the problem on uniform parallel machines. Schuurman and Vredeveld describe the so-called JUMP-algorithm, which is an extension of the 0/1-INTERCHANGE procedure to this machine environment. They define the *slack of a machine* as the maximum amount of processing requirement that

can be added to the machine without increasing the makespan of the schedule: $\Delta_i = s_i(L_{\max} - L_i)$, for $i = 1, \dots, m$. The maximum slack is denoted by $\Delta = \max_{1 \leq i \leq m} \Delta_i$. A *qualifying job* in this setting is a job j on a critical machine with $p_j < \Delta$. The JUMP-procedure iteratively moves a qualifying job to a machine with maximum slack, until there are no qualifying jobs left. Note that in the case of identical parallel machines, i.e., $s_i = 1$ for $i = 1, \dots, m$, $\Delta = L_{\max} - L_{\min}$ and this procedure is the same as 0/1-INTERCHANGE. A solution found by this algorithm has makespan that is at most $\frac{1-\sqrt{4m-3}}{2}$ times the optimum; the procedure terminates in $\mathcal{O}(n^2m)$ iterations, see Schuurman and Vredeveld (2001). We show that if the largest qualifying job is selected to move, then the JUMP-algorithm finds a jump-optimal solution in $\mathcal{O}(nm)$ iterations.

2 Lower bound on number of iterations

In this section, we provide an example for which 0/1-INTERCHANGE needs $\Omega(n^2)$ iterations to terminate.

We consider the following instance for the problem on two identical parallel machines. Let q be a positive integer. Then the number of jobs is $n = 3q + 1$ and the processing requirements are given by $1, 2, \dots, 2^q$, each length occurring three times, except for 2^q which has multiplicity one. For given q , we denote the schedule after the i th iteration of the 0/1-INTERCHANGE procedure by $S_i^{(q)}$. We represent a schedule by giving the processing requirements on both machines. Our initial schedule is given by

$$S_0^{(q)} \quad \begin{array}{l} M_1 : 1, \dots, 2^{q-1}, 1, \dots, 2^q \\ M_2 : 1, \dots, 2^{q-1} \end{array}$$

Lemma 1 *Let q be a positive integer. There exists a job selection rule such that the 0/1-INTERCHANGE procedure starting from $S_0^{(q)}$ terminates after $q(q+1)/2$ iterations.*

PROOF. We prove this lemma by induction on q . For $q = 1$, in the initial schedule M_2 processes one job of length 1 and all other jobs are scheduled on M_1 and these jobs are all qualifying jobs. After moving one job from M_1 to M_2 , the difference in machine loads is equal to 1 and there are no qualifying jobs.

Assume the claim is true for $q - 1$. In iteration i , for $i = 1, \dots, q - 1$, we select a job with processing requirement 2^{i-1} .

$$S_i^{(q)} \quad \begin{array}{l} M_1 : 1, \dots, 2^{q-1}, 2^i, \dots, 2^q \\ M_2 : 1, \dots, 2^{q-1}, 1, \dots, 2^{i-1} \end{array}$$

After $q - 1$ iterations, we arrive in schedule $S_{q-1}^{(q)}$.

$$S_{q-1}^{(q)} \quad \begin{array}{l} M_1 : 1, \dots, 2^{q-1}, 2^{q-1}, 2^q \\ M_2 : 1, \dots, 2^{q-1}, 1, \dots, 2^{q-2} \end{array}$$

The difference in machine loads in this schedule is $2^q + 1$ and we select the qualifying job of length 2^q to move in the next iteration.

$$S_q^{(q)} \quad \begin{array}{l} M_1 : 1, \dots, 2^{q-1}, 2^{q-1} \\ M_2 : 1, \dots, 2^{q-1}, 1, \dots, 2^{q-2}, 2^q \end{array}$$

This schedule can be written as

$$S_q^{(q)} \quad \begin{array}{l} M_1 : 2^{q-1}, 2^{q-1}, 1, \dots, 2^{q-2} \\ M_2 : 2^q, 1, \dots, 2^{q-2}, 1, \dots, 2^{q-1} \end{array}$$

This schedule is the schedule $S_0^{(q-1)}$ in which the numbering of the machines is reversed and two jobs of length 2^{q-1} are added to M_1 and one job of length 2^q is added to M_2 . By our induction hypothesis, we know that there exists a job selection rule such that 0/1-INTERCHANGE starting from this schedule needs $q(q-1)/2$ iterations to terminate. Hence, the total number of iterations is $q + q(q-1)/2 = q(q+1)/2$. \square

As $n = 3q + 1$, the following theorem is immediate.

Theorem 1 *For the problem of minimizing the makespan on identical parallel machines, 0/1-INTERCHANGE needs $\Omega(n^2)$ iterations in the worst case to find a jump-optimal solution.* \square

To show that the implementation of 0/1-INTERCHANGE proposed by Finn and Horowitz needs $\Omega(n^2)$ iterations to terminate, we use the same instance and the starting solution has the same assignment of the jobs to the machines. We assume w.l.o.g. that q is a multiple of three. We need to specify the order in which the jobs appear in the lists of the two machines. Hereto, we partition the set of jobs into three disjoint subsets, A_q , B_q , and C_q . The job of length 2^q is assigned to A_q ; each other job length occurs exactly once in a set. The order in which the jobs appear in the three sets is as follows.

$$\begin{array}{l} A_q : 2, 1, 2^3, 2^2, 2^4, \dots, 2^{3i}, 2^{3i-1}, 2^{3i+1}, \dots, 2^q, 2^{q-1} \quad (i = 2, \dots, \frac{q}{3} - 1) \\ B_q : 1, 2^2, 2^1, 2^3, \dots, 2^{3i-1}, 2^{3i-2}, 2^{3i}, \dots, 2^{q-1}, 2^{q-2} \quad (i = 2, \dots, \frac{q}{3} - 1) \\ C_q : 4, 2, 1, \dots, 2^{3i+2}, 2^{3i+1}, 2^{3i}, \dots, 2^{q-1}, 2^{q-2}, 2^{q-3} \quad (i = 1, \dots, \frac{q}{3} - 2) \end{array}$$

The initial schedule is given by

$$S_0^{(q)} \quad \begin{array}{l} M_1 : A_q, C_q \\ M_2 : B_q \end{array}$$

Lemma 2 *Let q be a positive multiple of three. Using the job selection rule proposed by Finn and Horowitz, 0/1-INTERCHANGE starting from $S_0^{(q)}$ needs $q(q+1)/2$ iterations to terminate.*

PROOF. We prove this lemma by induction on q . For $q = 3$, it is easy, but tedious, to see that the algorithm takes six iterations.

Assume the claim is true for $q - 3$. In the first q iterations only jobs from the set A_q are moved and we end up in the following schedule.

$$S_q^{(q)} \quad \begin{array}{l} M_1 : 2^{q-1}, C_q \\ M_2 : B_q, A_{q-3}, 2^{q-2}, 2^q \end{array}$$

In the next $q - 1$ iterations, jobs from the set B_q are moved and we obtain the schedule $S_{2q-1}^{(q)}$.

$$S_{2q-1}^{(q)} \quad \begin{array}{l} M_1 : 2^{q-1}, C_q, B_{q-3}, 2^{q-3}, 2^{q-1} \\ M_2 : 2^{q-2}, A_{q-3}, 2^{q-2}, 2^q \end{array}$$

In the next iteration, the first job to be scanned is the first one of length 2^{q-1} on machine M_1 . As the difference in machine loads in the current schedule is $2^{q-1} - 1$, this job is not a qualifying job. Hence, the procedure moves in the next $q - 2$ iterations only jobs from the set C_q and we find the schedule $S_{3q-3}^{(q)}$.

$$S_{3q-3}^{(q)} \quad \begin{array}{l} M_1 : 2^{q-1}, 2^{q-1}, 2^{q-3}, B_{q-3}, 2^{q-3}, 2^{q-1} \\ M_2 : 2^{q-2}, A_{q-3}, 2^{q-2}, 2^q, C_{q-3}, 2^{q-2} \end{array}$$

This schedule is equal to $S_0^{(q-3)}$ in which the numbering of the machines is reversed and some jobs have been added to both machines. We claim that these additional jobs do not become qualifying jobs. Hence, 0/1-INTERCHANGE continues as if it was starting from schedule $S_0^{(q-3)}$. By our induction hypothesis, we know that the procedure still needs $(q - 3)(q - 2)/2$ iterations to terminate and the total number of iterations is $(q - 3)(q - 2)/2 + 3q - 3 = q(q + 1)/2$.

To prove the claim that the additional jobs in schedule $S_{3q-3}^{(q)}$ do not become qualifying jobs, note that the difference in machine loads in this schedule is $2^{q-2} - 1$. Hence, all additional jobs on the critical machine M_2 are too large to be qualifying jobs. M_1 becomes the critical machine after $q - 3$ iterations in which only jobs from A_{q-3} have been moved. In the resulting schedule, the difference in machine loads is $2^{q-3} - 1$ and the jobs with processing requirements of at least 2^{q-3} cannot become qualifying jobs. \square

The following theorem is a direct consequence of this lemma.

Theorem 2 *For the problem of minimizing makespan on identical parallel machines, 0/1-INTERCHANGE, using the job selection rule proposed by Finn and Horowitz, needs $\Omega(n^2)$ iterations in the worst case to find a jump-optimal solution. \square*

3 Upper bound on the number of iterations

We now turn to the uniform parallel machine environment. As mentioned in the introduction, Schuurman and Vredeveld proposed the so-called JUMP-procedure, which is a generalization of 0/1-INTERCHANGE. In Figure 2 we give

```

procedure JUMP
  initialize  $S$  by placing the  $n$  jobs on the  $m$  machines in any order
  while there is a job  $j \in J_{\max}$  with  $p_j < \Delta$ 
    Let job  $j$  be such that  $p_j = \max\{p_k : k \in J_{\max}, p_k < \Delta\}$ 
    modify  $S$  by moving job  $j$  to a machine  $M_i$  with  $\Delta_i = \Delta$ 
  end while

```

Figure 2: The JUMP-algorithm

a formal description of the JUMP-algorithm in which the largest qualifying job is selected to move, until there are no qualifying jobs left. Recall that for this setting a qualifying job is a job j on a critical machine with $p_j < \Delta$.

Theorem 3 *For the problem of minimizing the makespan on uniform parallel machines, the JUMP-algorithm as given in Figure 2 terminates in $\mathcal{O}(nm)$ iterations.*

PROOF. The JUMP-algorithm computes a sequence of schedules with non-increasing makespan and maximum slack. It finishes when all jobs on the critical machines have processing requirements more than or equal to the maximum slack: $p_j \geq \Delta$. We show that once a job is moved away from a machine, it never returns to this machine. Hence, a job can move at most $m - 1$ times and the total number of iterations is bounded by $n(m - 1)$.

We denote the values of J_i , L_i , L_{\max} , Δ_i , and Δ at the beginning of iteration t by $J_i(t)$, $L_i(t)$, $L_{\max}(t)$, $\Delta_i(t)$, and $\Delta(t)$. Suppose a job j moves from machine M_i in iteration t_0 . If after t_0 no job moves to machine M_i , then, in particular, job j does not move to machine M_i . Otherwise, let t_2 be the first iteration after t_0 in which M_i is a machine with maximum slack. Then there exists an iteration $t_1 \in [t_0, t_2)$ such that M_i is critical in iteration t_1 and not critical in iterations $t \in (t_1, t_2)$. Let job k be the job that is moved in iteration t_1 . Then $L_i(t_2) = L_i(t_1) - p_k/s_i$. By monotonicity of $L_{\max}(t)$, we have that $\Delta(t_2) = s_i(L_{\max}(t_2) - L_i(t_2)) \leq s_i(L_{\max}(t_1) - L_i(t_1)) + p_k = p_k$. Hence, in iteration t_2 and later iterations, job k is too large to be a qualifying job. As M_i is not a machine with maximum slack in iterations $t \in [t_0, t_1]$, the jobs assigned to M_i in iteration t_1 were also scheduled on M_i in iteration t_0 . By our job selection rule and the fact that $\Delta(t_0) \geq \Delta(t_1)$, we know that $p_j \geq p_k$ and therefore $p_j \geq \Delta(t_2)$. Hence, job j cannot return to machine M_i . \square

References

P. BRUCKER, J. HURINK, AND F. WERNER (1997). Improving local search heuristics for some scheduling problems II. *Discrete Applied Mathematics* **72**, 47–69.

G. FINN AND E. HOROWITZ (1979). A linear time approximation algorithm for multiprocessor scheduling. *BIT* **19**, 312–320.

J.K. LENSTRA, A.H.G. RINNOOY KAN, AND P. BRUCKER (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics* **1**, 343–362.

P. SCHURMAN AND T. VREDEVELD (2001). Performance guarantees of local search for multiprocessor scheduling, Manuscript. Preliminary version in *Proceedings of 8th Integer Programming and Combinatorial Optimization Conference*, LNCS 2081, 370–382, Springer, Berlin.