
Department of Applied Mathematics
Faculty of EEMCS



University of Twente
The Netherlands

P.O. Box 217
7500 AE Enschede
The Netherlands

Phone: +31-53-4893400

Fax: +31-53-4893114

Email: memo@math.utwente.nl
www.math.utwente.nl/publications

Memorandum No. 1801

**Very large-scale neighborhoods with
performance guarantees for minimizing
makespan on parallel machines**

T. BRUEGGEMANN, J.L. HURINK, T. VREDEVELD¹
AND G.J. WOEGINGER²

May, 2006

ISSN 0169-2690

¹Department of Quantitative Economics, Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands

²Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Very Large-Scale Neighborhoods with Performance Guarantees for Minimizing Makespan on Parallel Machines

Tobias Brueggemann^{1,*}, Johann L. Hurink²,

*Department of Applied Mathematics, University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands*

Tjark Vredeveld,

*Department of Quantitative Economics, Maastricht University,
P.O. Box 616, 6200 MD Maastricht, The Netherlands*

Gerhard J. Woeginger².

*Dept. of Math. and Comp. Sci., Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

Abstract

We study the problem of minimizing the makespan on m parallel machines. We introduce a very large-scale neighborhood of exponential size (in the number of machines) that is based on a matching in a complete graph. The idea is to partition the jobs assigned to the same machine into two sets. This partitioning is done for every machine with some chosen rule to receive $2m$ parts. A new assignment is received by putting to every machine exactly two parts. The neighborhood \mathcal{N}_{split} consists of all possible rearrangements of the parts to the machines. The best assignment of \mathcal{N}_{split} can be calculated in time $\mathcal{O}(m \log m)$ by determining the perfect matching having minimum maximal edge weight in an improvement graph, where the vertices correspond to parts and the weights on the edges correspond to the sum of the processing times of the jobs belonging to the parts. Additionally, we examine local optima in this neighborhood and in combinations with other neighborhoods. We derive performance guarantees for these local optima.

Key words: local search, performance guarantee, parallel machines, makespan
MSC2000: 90B35, 68W25

1 Introduction

In this paper, we consider the following multiprocessor scheduling problem. Given are n jobs each of which has to be scheduled on one of m identical parallel machines. The time it takes for a job j to be fully processed is denoted by p_j . A machine can process at most one job at a time, and a job may not be preempted. The goal is to schedule the jobs in such a way that the *makespan* is minimized, i.e., we want the last job to complete as early as possible. In the standard notation of [8], this problem is denoted as $P||C_{\max}$.

This problem is known to be strongly NP-hard for m being part of the input [7]. Therefore, we search for approximate solutions. If an algorithm is guaranteed to deliver a solution that has value at most ρ times the optimal solution value, we call it a ρ -*approximation* algorithm; the value ρ is called the (*worst-case*) *performance guarantee*. A well known approximation algorithm for the problem under consideration is the *LPT-algorithm* due to Graham [9]: starting from an empty schedule, we select the job with longest processing time among the unscheduled jobs and schedule this job on the machine with currently minimal workload. This LPT-algorithm has a performance guarantee of $4/3 - 1/3m$.

Another way to find approximate solutions is through *local search*, see e.g. [1]. These methods iteratively search through the set of feasible solutions. Starting from an initial solution, a local search procedure moves from one feasible solution to a neighboring one until some stopping criteria are met. The choice of a suitable neighborhood function has an important influence on the performance of local search.

The simplest form of local search is *iterative improvement*, also called local improvement or, in the case of minimization problems, descent algorithms. This method iteratively chooses a better solution in the neighborhood of the current one, and it stops when no better solution is found. The final solution is called a *local optimum*.

Recently, there has been an increasing interest in the quality of local optima and the time needed to obtain these local optima through iterative improve-

* Corresponding author.

Email addresses: `t.brueggemann@math.utwente.nl` (Tobias Brueggemann), `j.l.hurink@math.utwente.nl` (Johann L. Hurink), `t.vredeveld@ke.unimaas.nl` (Tjark Vredeveld), `gwoegi@win.tue.nl` (Gerhard J. Woeginger).

¹ supported by the Netherlands Organization for Scientific Research (NWO) grant 613.000.225 (Local Search with Exponential Neighborhoods).

² supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society).

ment. For the parallel machine scheduling problem under consideration, Finn and Horowitz [6] showed that a so-called move-optimal solution is guaranteed to deliver a solution with value no more than $2 - 2/(m + 1)$ times the optimal makespan. Moreover, this bound is tight [11]. Brucker et al. [3] showed that the iterative improvement procedures needs $\mathcal{O}(n^2)$ moves to come to local optimal solution, and this bound is tight [10]. For performance guarantees of local search methods regarding makespan minimization, we refer to [11,10]. If the objective is to minimize total weighted completion time, Brueggemann et al. [4] gave a performance guarantee of $3/2 - 1/2m$ for move-optimal schedules.

Over the last years, very large-scale neighborhoods have received considerable attention. These neighborhoods mostly contain up to an exponential number of solutions, but allow a polynomial exploration. A survey about very large-scale neighborhood techniques is given by Ahuja et al. [2] and Deĭneko and Woeginger [5] present an overview of very large-scale neighborhoods for the traveling salesman and quadratic assignment problem.

In Section 2, we define a very large-scale neighborhood, called *split-neighborhood* and in the following sections we investigate its worst-case behavior. In Section 3, we see that a split-optimal solution has the same performance guarantee as the simple move-optimal solutions. In Sections 4 and 5, we give performance guarantees on combined move-optimal and split-optimal solutions. If we combine the two neighborhoods in the most straightforward way, we see that the performance guarantee marginally improves but is still essentially 2, whereas a better combination leads to a performance guarantee of $3/2$.

2 Neighborhoods

As mentioned in the introduction, an important part of local search algorithms is the definition of the neighborhood on which the method operates. Before discussing the neighborhoods, we first describe our representation of a schedule. As the sequence in which the jobs are processed does not influence the makespan of a schedule for a given assignment, we represent a schedule by such an assignment of jobs to machines, $A : J \rightarrow \{1, \dots, m\}$, where $J = \{1, \dots, n\}$ is the set of jobs. This is equivalent to a partitioning of the set of jobs into m disjoint subsets M_1^A, \dots, M_m^A , where $M_i^A = \{j \in J : A(j) = i\}$ is the set of jobs scheduled on machine i . If there is no ambiguity, we write M_i for M_i^A . The *workload* of machine i is denoted by

$$L_i^A = \sum_{j \in M_i} p_j,$$

and this workload is equal to the completion time of the last job scheduled on machine i . Again, if there is no ambiguity, we write L_i for L_i^A . Hence, for a given assignment A of jobs to machines, the makespan is equal to the machine with maximum workload:

$$C_{\max}^A = \max_i L_i^A.$$

We call such a machine with maximum workload a *critical machine*.

The move-neighborhood Probably the most basic neighborhood is the *move-neighborhood*. A neighbor is obtained by moving one job from its current machine to another machine. More formally, let j be the selected job and i be the selected machine. Then the neighbor obtained by moving job j to machine i is the assignment

$$A_{\text{move}}^{(j,i)}(k) = \begin{cases} A(k) & \text{if } k \neq j, \\ i & \text{if } k = j. \end{cases}$$

Note that $A_{\text{move}}^{(j,i)}$ only differs from A if $A(j) \neq i$. The neighborhood $\mathcal{N}_{\text{move}}(A)$ of an assignment A contains all assignments that can be obtained by moving one job to some machine, i.e.,

$$\mathcal{N}_{\text{move}}(A) = \left\{ A_{\text{move}}^{(j,i)} : j \in J \text{ and } i = 1, \dots, m \right\}.$$

We call an assignment A *move-optimal* if for all $A' \in \mathcal{N}_{\text{move}}(A)$, $C_{\max}^A \leq C_{\max}^{A'}$ and, in case of $C_{\max}^A = C_{\max}^{A'}$, the number of critical machines in A is at most the number of critical machines in A' . Finn and Horowitz [6] gave the following upper bound on the performance guarantee of move-optimal assignments.

Theorem 1 *Let A be a move-optimal assignment, and let C_{\max}^* denote the optimal makespan. Moreover, let $n_k = \max\{|M_i| : L_i = C_{\max}^A\}$, the (maximum) number of jobs on a critical machine in the assignment A . Then*

$$C_{\max}^A \leq \frac{n_k m}{(n_k - 1)m + 1} C_{\max}^*.$$

Moreover, if $n_k = 1$, then $C_{\max}^A = C_{\max}^*$.

The bound in Theorem 1 attains its maximum for $n_k = 2$, yielding a performance guarantee of $2 - 2/(m + 1)$. This bound has been proven tight by Schuurman and Vredeveld [11], see Figure 1 for the assignments attaining this bound.

The split-neighborhood The *split-neighborhood* is of exponential size in the number of machines. It bases on an operator that partitions the set of jobs assigned to a machine i into two disjoint sets. A *split-operator* $\text{split} : \mathcal{P}(J) \rightarrow$

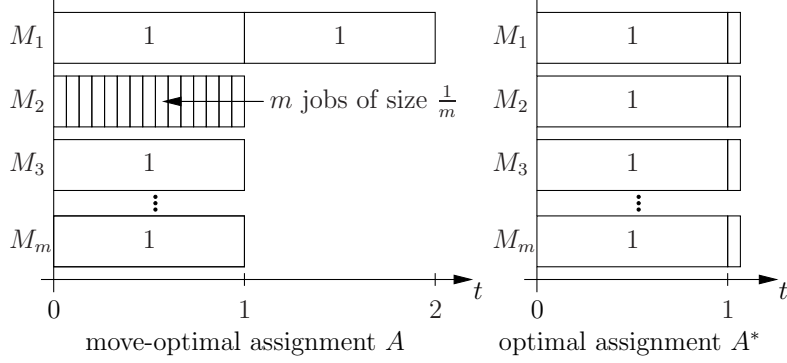


Fig. 1. Worst-case move-optimal schedule

$\mathcal{P}(J) \times \mathcal{P}(J)$ applied to a set M_i of jobs produces $\text{split}(M_i) = (M_{i1}, M_{i2})$ and has the following properties:

- the sets M_{i1} and M_{i2} are a partition of M_i ,
- for the workloads of the sets holds

$$L_{i1} = \sum_{j \in M_{i1}} p_j \geq \sum_{j \in M_{i2}} p_j = L_{i2}.$$

We refer to the sets M_{i1} and M_{i2} of a machine i (or a set M_i) as the *left part* and *right part*, respectively.

If we use the split-operator on all sets M_i given by an assignment A , we obtain $2m$ parts M_{i1} and M_{i2} for $i = 1, \dots, m$. Abusing notation, we denote the set of these $2m$ parts by

$$\text{split}(A) = \{ M_{i1}, M_{i2} : \text{split}(M_i) = (M_{i1}, M_{i2}) \text{ for } i = 1, \dots, m \}.$$

We call an assignment A' a *neighbor* of A , if A' can be received by assigning the jobs of exactly two of the $2m$ parts from $\text{split}(A)$ to each machine. The neighborhood $\mathcal{N}_{\text{split}}$ of an assignment A is defined as

$$\mathcal{N}_{\text{split}}(A) := \{ A' : A' \text{ is neighbor of } A \}.$$

We are interested in the *best neighbor* A_1 of the neighborhood $\mathcal{N}_{\text{split}}(A)$, i.e. we are seeking an assignment $A_1 \in \mathcal{N}_{\text{split}}(A)$ with

$$C_{\max}^{A_1} \leq C_{\max}^{A'} \text{ for all } A' \in \mathcal{N}_{\text{split}}(A),$$

and among all assignments $A' \in \mathcal{N}_{\text{split}}(A)$ with $C_{\max}^{A_1} = C_{\max}^{A'}$, A_1 is the assignment with minimal number of machines attaining the makespan.

Although the size of the neighborhood is exponentially large in the number of machines, the following fact tells us that the best neighbor can be found in $\mathcal{O}(m \log m)$ time.

Fact 2 Given $2m$ numbers $a_1 \geq \dots \geq a_{2m}$. A perfect matching of these numbers such that the maximum of the sum of the two numbers of a pair is minimized, is obtained by matching a_i to a_{2m+1-i} . Moreover, this matching minimizes the number of pairs whose sum equals this maximum.

In other words, an optimal solution for the bottleneck assignment problem is obtained by ordering the cost-matrix of the assignment problem, so that it fulfills the bottleneck Monge property.

Thus, we obtain the best neighbor of the neighborhood $\mathcal{N}_{split}(A)$ by rearranging the $2m$ parts T_1, \dots, T_{2m} obtained from $split(A)$, so that for the sum of processing times of the parts holds $L_{T_1} \geq \dots \geq L_{T_{2m}}$ and the jobs of the parts T_i and T_{2m+1-i} are assigned to machine i for $i = 1, \dots, 2m$.

We call an assignment *split-optimal* if for all $A' \in \mathcal{N}_{split}(A)$, $C_{\max}^A \leq C_{\max}^{A'}$ and the number of critical machines in A is at most the number of critical machines in A' .

Combinations of move and split-neighborhood As we will see in the following section, a split-optimal assignment needs not to be a move-optimal. Hence, we also consider assignments, that are both, move-optimal and split-optimal. These local optima may however be improved by moving one job neither increasing nor decreasing the makespan, and then find a better split-neighbor. Therefore, we define a *lexicographic-move-optimal* assignment. For a given assignment A and $A' \in \mathcal{N}_{move}(A)$, we reorder the machines in A and A' so that

$$\begin{aligned} L_1^A &\geq \dots \geq L_m^A \quad \text{and} \\ L_1^{A'} &\geq \dots \geq L_m^{A'}. \end{aligned}$$

The assignment A' is called *lexicographically better* than A , if there exists a machine k such that

$$\begin{aligned} L_i^{A'} &= L_i^A \quad \text{for } i = 1, \dots, k-1, \\ L_k^{A'} &< L_k^A. \end{aligned} \tag{1}$$

We say that A is *lexicographic-move-optimal*, or *lexmove-optimal*, if there exists no move-neighbor $A' \in \mathcal{N}_{move}(A)$ that is lexicographically better than A .

Note that the move-optimal assignment A in Figure 1 is also lexmove-optimal. Therefore, the move-optimal and the lexmove-optimal assignments have the same performance guarantee. As will be seen in the following, the performance guarantee of an assignment that is both, lexmove-optimal and split-optimal, is better than that of a move and split-optimal assignment.

3 Performance guarantee on split-optimal assignments

The quality of split-optimal assignments depends on the split-operator. In this paper, we consider only split-operators that obtain move-optimal partitions, i.e., if $\text{split}(M_i) = (M_{i1}, M_{i2})$, with $L_{i1} \geq L_{i2}$, then for all $j \in M_{i1}$, we have $L_{i2} + p_j \geq L_{i1}$. We call such a split-operator a *move-optimal split-operator*.

For a split-optimal assignment A , despite the fact that $L_{i1} \geq L_{i2}$ for all machines $i = 1, \dots, m$, for a critical machine k we have the following properties for any machine i :

$$\begin{aligned} &\text{if } L_{i1} \geq L_{k1} \text{ then } L_{i2} \leq L_{k2} \text{ holds,} \\ &\text{if } L_{i1} < L_{k1} \text{ then } L_{i2} \geq L_{k2} \text{ holds.} \end{aligned} \tag{2}$$

The first statement follows from the fact that k is a critical machine, and the second statement holds since A is split-optimal.

The performance guarantee of a split-optimal assignment, using a move-optimal split-operator, does not improve on the bound obtained by move-optimal assignment.

Theorem 3 *Let A be a split-optimal assignment using a move-optimal split-operator. Then the makespan of A is bounded by $C_{\max}^A \leq (2 - \frac{2}{m+1})C_{\max}^*$, where C_{\max}^* denotes the value of the optimal makespan.*

PROOF. W.l.o.g. we assume that $C_{\max}^A = 1$. Let k be a critical machine, i.e. $L_k = 1$. If $\sum_j p_j \geq mL_{k1} + L_{k2}$, then the optimal makespan can be bounded from below by $C_{\max}^* \geq \frac{1}{m} \sum_j p_j \geq L_{k1} + L_{k2}/m$. Using the fact that $L_{k1} + L_{k2} = 1$, we have

$$\frac{C_{\max}^A}{C_{\max}^*} \leq \frac{m}{(m-1)L_{k1} + 1} \leq \frac{2m}{m+1} = 2 - \frac{2}{m+1},$$

as the above expression is maximized for minimal L_{k1} and by $L_{k1} \geq L_{k2}$ we know that $L_{k1} \geq 1/2$.

On the other hand, if $\sum_j p_j < mL_{k1} + L_{k2}$, then a machine l with minimal load satisfies

$$L_l \leq \sum_{i \neq k} L_i / (m-1) < L_{k1}. \tag{3}$$

Moreover, by (2), we know that (3) implies $L_{l2} \geq L_{k2}$. Hence,

$$L_{k1} > L_l \geq 2L_{l2} \geq 2L_{k2}. \tag{4}$$

From the fact that a move-optimal split-operator is used to obtain the sets M_{i1} and M_{i2} , we know that for all $j \in M_{k1}$, $L_{k2} + p_j \geq L_{k1}$. Therefore, from (4)

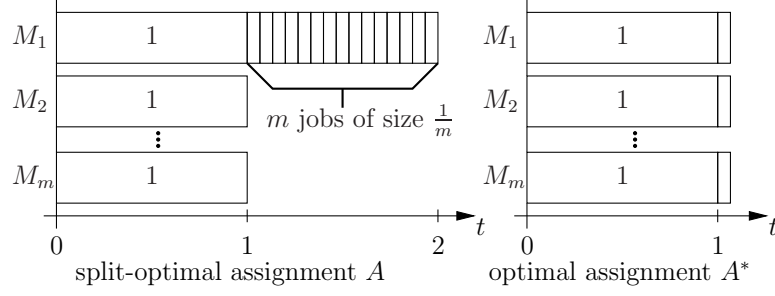


Fig. 2. Worst-case example for split-optimal assignments

it follows that $p_j > \frac{1}{2}L_{k1}$ for all $j \in M_{i1}$. Hence, M_{i1} contains only one job and

$$C_{\max}^A = L_k = L_{k1} + L_{k2} < \frac{3}{2}L_{k1} \leq \frac{3}{2}C_{\max}^*,$$

as $C_{\max}^* \geq p_j$ for all $j \in J$ and thus $C_{\max}^* \geq L_{k1}$.

For $m \geq 3$, the theorem is proven, as $\frac{3}{2} \leq 2m/(m+1)$. For $m = 2$, it follows from (4) that

$$C_{\max}^* \geq \frac{1}{2} \sum_j p_j \geq \frac{1}{2}(L_k + 2L_{l2}) \geq \frac{1}{2}(L_{k1} + 3L_{k2}) = \frac{3}{2} - L_{k1},$$

where the last equality follows from the fact that $L_{k2} = 1 - L_{k1}$. Moreover, as $C_{\max}^* \geq L_{k1}$, due to the fact that M_{k1} contains only one job, we have

$$C_{\max}^* \geq \max\{L_{k1}, \frac{3}{2} - L_{k1}\},$$

which is minimal for $L_{k1} = \frac{3}{4}$. Therefore, for $m = 2$, we have

$$C_{\max}^A \leq \frac{4}{3}C_{\max}^*.$$

□

To show that the analysis is tight, consider the following instance consisting of m jobs with processing time 1 and m jobs with processing time $1/m$. In the split-optimal assignment A , we schedule on every machine one job with processing time 1 and on the first machine all jobs with processing time $1/m$ are scheduled. It is easy to check that this assignment is split-optimal for a move-optimal split-operator and it has makespan $C_{\max}^A = 2$. In an optimal assignment A^* , we schedule on every machine one job with processing time 1 and one with processing time $1/m$. The optimal makespan is $C_{\max}^* = 1 + 1/m$, and thus $C_{\max}^A = 2m/(m+1)C_{\max}^*$. See Figure 2 for an illustration.

4 Split-optimal and move-optimal assignments

The worst-case instance for split-optimal assignments, showing the tightness of our analysis in the previous section, is obviously not move-optimal. This raises the question whether a combination of the two neighborhoods gives a better performance guarantee, which is answered in Theorem 9, for move-optimal split-operators. Moreover, if the total processing time of the jobs compared to the load of the left part of a critical machine is small, we can even prove that this move-optimal and split-optimal assignment is globally optimal.

Like in the previous section, we assume w.l.o.g. that property (2) holds for a local optimal assignment A and that $C_{\max}^A = 1$.

Lemma 4 *Let A be a move-optimal and split-optimal assignment using a move-optimal split-operator. If there exists a critical machine k with*

$$\sum_j p_j < mL_{k1} + L_{k2},$$

then $C_{\max}^A = C_{\max}^$, where C_{\max}^* denotes the optimal makespan.*

PROOF. Let l be a machine with minimal load. Then, we know by (3) that $L_l < L_{k1}$. By move-optimality of the assignment A , we know that for any job $j \in M_k$

$$L_l + p_j \geq L_k = L_{k1} + L_{k2}.$$

Hence, $p_j > L_{k2}$ for $j \in M_k$, and thus M_{k2} contains no job at all, i.e., $L_{k2} = 0$. It follows from the move-optimal split-operator, that whenever M_{k2} is empty, M_{k1} contains only one job, j_1 . Hence, $C_{\max}^A = L_k = p_{j_1} \leq C_{\max}^*$. \square

From this lemma, it follows that we only have to consider cases in which the total load on all machines is large enough. Moreover, if L_{k1} is large enough, we can actually prove a bound on the makespan of this local optimal assignment, which is better than the guarantee in Theorem 9.

Lemma 5 *Let A be a move-optimal and split-optimal assignment obtained by using a move-optimal split-operator. If there exists a critical machine k , satisfying $L_{k1} \geq \frac{2}{3}$, then the makespan of A can be bounded by*

$$C_{\max}^A \leq \frac{3m}{2m+1} C_{\max}^*,$$

where C_{\max}^ denotes the optimal makespan.*

PROOF. By Lemma 4, we only have to consider the case that $\sum_j p_j \geq mL_{k_1} + L_{k_2}$. Hence, the optimal makespan can be bounded from below by $C_{\max}^* \geq \frac{1}{m} \sum_j p_j \geq \frac{(m-1)L_{k_1} + 1}{m}$. As, $C_{\max}^A = 1$, we thus have

$$\frac{C_{\max}^A}{C_{\max}^*} \leq \frac{m}{(m-1)L_{k_1} + 1} \leq \frac{3m}{2m+1},$$

where the last inequality is due to $L_{k_1} \geq 2/3$. \square

Let k be a critical machine. Before we prove the performance guarantee on a move-optimal and split-optimal assignment A , we first partition the set of machines into several subsets, based on the properties of the machines in such an assignment.

$$\begin{aligned} S_{<} &= \{i : L_{i1} < L_{k1}\}, \\ S_{\geq} &= \{i : L_{i1} \geq L_{k1}\}, \\ S_{\text{multi}} &= \{i \in S_{\geq} : |M_{i1}| \geq 2\}, \\ S_{\text{single}} &= S_{\geq} \setminus (S_{\text{multi}} \cup \{k\}). \end{aligned} \tag{5}$$

That is, $S_{<}$ is the set of machines that have a left part which is smaller than L_{k1} , and S_{\geq} is the set of the remaining machines. This set is again partitioned in one set containing all machines that have at least two jobs in the left part and the remaining machines in $S_{\geq} \setminus \{k\}$ containing exactly one job in the left part. Note that, $S_{\geq} \setminus \{k\} = S_{\text{multi}} \cup S_{\text{single}}$.

The load of a machine in each of the above classes, we can bound as follows.

Lemma 6 *Let A be a move-optimal and split-optimal schedule, for a move-optimal split-operator and let k be a critical machine in this assignment. Moreover, let $S_{<}$ and S_{multi} be as defined in (5). Then:*

$$\begin{aligned} L_i &\geq 2(1 - L_{k1}) && \text{for } i \in S_{<}, \\ L_i &\geq \frac{3}{2}L_{k1} && \text{for } i \in S_{\text{multi}}. \end{aligned}$$

PROOF. Consider a machine $i \in S_{<}$. Then by property (2), we know that $L_{i1} < L_{k1}$ implies that $L_{i2} \geq L_{k2}$. Moreover, as $L_{i1} \geq L_{i2}$, we have that $L_i \geq 2L_{i2} \geq 2L_{k2} = 2(1 - L_{k1})$.

Now, let $i \in S_{\text{multi}}$, and let $j_s \in M_{i1}$ be the smallest job in the left part of machine i . As M_{i1} contains at least two jobs, we know that $p_{j_s} \leq \frac{1}{2}L_{i1}$. Due to the move-optimality of the split-operator, we also know that $L_{i2} \geq L_{i1} - p_{j_s} \geq \frac{1}{2}L_{i1}$. Hence, $L_i \geq \frac{3}{2}L_{i1} \geq \frac{3}{2}L_{k1}$. \square

Lemma 7 *Let A be a move-optimal and split-optimal assignment for a move-optimal split-operator and k be a critical machine in A . Moreover, let $S_{<}$ be as defined in (5). If $1/2 \leq L_{k1} \leq 2/3$ and $|S_{<}| \geq 1$, then*

$$\frac{C_{\max}^A}{C_{\max}^*} \leq \begin{cases} \frac{2m}{m+2} & \text{for } m \geq 4, \\ \frac{3m}{2m+1} & \text{for } m \leq 3, \end{cases}$$

where C_{\max}^* denotes the optimal makespan.

PROOF. Using Lemma 6, we can bound the optimal makespan by

$$\begin{aligned} mC_{\max}^* &\geq \sum_j p_j \geq 1 + 2|S_{<}|(1 - L_{k1}) + (m - 1 - |S_{<}|)L_{k1} \\ &\geq 1 + (2 - 3L_{k1})|S_{<}| + (m - 1)L_{k1} \geq 3 + (m - 4)L_{k1}, \end{aligned} \quad (6)$$

where the last inequality is due to $L_{k1} \leq 2/3$. For $m \geq 4$, the expression in (6) is minimized for L_{k1} minimal, whereas for $m \leq 3$, it is minimized for L_{k1} maximal. Using the fact that $1/2 \leq L_{k1} \leq 2/3$ and $C_{\max}^A = 1$, we have

$$\frac{C_{\max}^A}{C_{\max}^*} \leq \begin{cases} \frac{2m}{m+2} & \text{for } m \geq 4, \\ \frac{3m}{2m+1} & \text{for } m \leq 3. \end{cases}$$

□

Lemma 8 *Let A be a move-optimal and split-optimal assignment for a move-optimal split-operator and k be a critical machine in A . Moreover, let S_{multi} be as defined in (5). If $1/2 \leq L_{k1} \leq 2/3$ and $|S_{\text{multi}}| \geq 2$, then*

$$\frac{C_{\max}^A}{C_{\max}^*} \leq \frac{2m}{m+2},$$

where C_{\max}^* denotes the optimal makespan.

PROOF. Consider a move-optimal and split-optimal assignment A for a move-optimal split-operator and let $S_{<}$, S_{multi} , and S_{single} be as defined in (5). For $L_{k1} \leq 2/3$, we know from Lemma 6 that for $i \in S_{<}$, $L_i \geq L_{k1}$. Hence, using Lemma 6, we can bound the optimal makespan by

$$C_{\max}^* \geq \frac{1 + (|S_{\text{multi}}|/2 + m - 1)L_{k1}}{m} \geq \frac{1 + mL_{k1}}{m} \geq \frac{2 + m}{2m},$$

where the second inequality is due to $|S_{\text{multi}}| \geq 2$ and the last is due to $L_{k1} \geq 1/2$. As by assumption $C_{\max}^A = 1$, the lemma is proven. □

Theorem 9 *A move-optimal and split-optimal assignment, obtained by a move-optimal split-operator, has a performance guarantee of $2 - \frac{4}{m+3}$.*

PROOF. Let A be a move-optimal and split-optimal assignment for a move-optimal split-operator and let k be a critical machine in A . Since $\frac{3m}{2m+1} \leq \frac{2m+2}{m+3}$, by Theorem 1 we only need to consider assignments A in which a critical machine k contains exactly two jobs. Moreover, by Lemma 5, we may restrict ourselves to the case that $L_{k1} \in [\frac{1}{2}, \frac{2}{3}]$, and by Lemma 4, we may assume that $\sum_j p_j \geq mL_{k1} + L_{k2}$.

As $\max\{\frac{2m}{m+2}, \frac{3m}{2m+1}\} \leq \frac{2m+2}{m+3}$, due to Lemma 7 we can restrict ourselves to the case that there is no machine i with $L_{i1} < L_{k1}$. Due to Lemma 8 we assume that there is at most one machine i with $|M_{i1}| \geq 2$. Note that if no such machine exists, there are m jobs of length at least L_{k1} and one job of length $1 - L_{k1} \leq L_{k1}$. Then, by the pigeonhole principle $C_{\max}^* = C_{\max}^A$. Hence, we assume that there is exactly one machine s with $|M_{s1}| \geq 2$.

Let j_1 be the smallest job in M_{s1} . If $p_{j_1} \geq \frac{m+3}{2m+2} - L_{k2}$, then there are $m - 1$ jobs of length L_{k1} , one job of length $L_{k2} = 1 - L_{k1} \leq L_{k1}$ and at least two jobs of length $\frac{m+3}{2m+2} - L_{k2}$, and by the pigeonhole principle, we know that $C_{\max}^* \geq \frac{m+3}{2m+2}$.

On the other hand, if $p_{j_1} \leq \frac{m+3}{2m+2} - (1 - L_{k1})$, we can bound the load of the right part of machine s by $L_{s2} \geq L_{s1} - p_{j_1}$. Hence, using the fact that $L_{k1} \geq 1/2$, we can bound the total workload by

$$\begin{aligned} \sum_j p_j &\geq L_s + \sum_{i \neq s} L_i \geq (m-1)L_{k1} + 1 - L_{k1} + L_{s1} + L_{s2} \\ &\geq (m-2)L_{k1} + 1 + 2L_{s1} - p_{j_1} \geq mL_{k1} + 1 - p_{j_1} \\ &\geq (m-1)L_{k1} + 2 - \frac{m+3}{2m+2} \geq \frac{m-1}{2} + 2 - \frac{m+3}{2m+2} \\ &= \frac{m^2 + 3m}{2m+2}. \end{aligned}$$

This implies that the optimal makespan can be bounded by

$$C_{\max}^* \geq \frac{1}{m} \sum_j p_j \geq \frac{m+3}{2m+2},$$

and thus we obtain

$$\frac{C_{\max}^A}{C_{\max}^*} \leq \frac{2m+2}{m+3}.$$

□

For instances with an odd number of machines, the analysis of the previous

theorem is tight. If we schedule m jobs of length 1 and m jobs of length $2/(m+1)$ as illustrated by the assignment A in Figure 3, we obtain a move-optimal and split-optimal assignment for a move-optimal split-operator, with makespan $C_{\max}^A = 2$. In the optimal schedule, all machines have the same workload and $C_{\max}^* = 1 + \frac{2}{m+1}$. For the split-optimality of this example, we

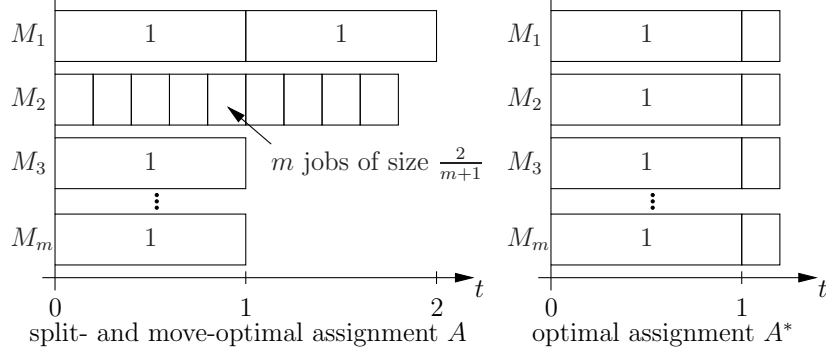


Fig. 3. Worst-case example for odd number of machines

need that the left part of machine M_2 has workload equal to 1. Therefore, this example only works for an odd number of machines. For even number of machines, a lower bound on the performance guarantee is $\frac{2m}{m+2}$. This bound is obtained by an instance with m jobs of size 1 and $m-1$ jobs of size $2/m$. In the move-optimal and split-optimal assignment, these jobs are scheduled similar as in Figure 3.

In the case of $m = 2$ machines, the bound cannot be improved either, as is shown by the example in Figure 4, which has makespan $C_{\max}^A = 6$ for the local optimum and $C_{\max}^* = 5$, in an optimal schedule.

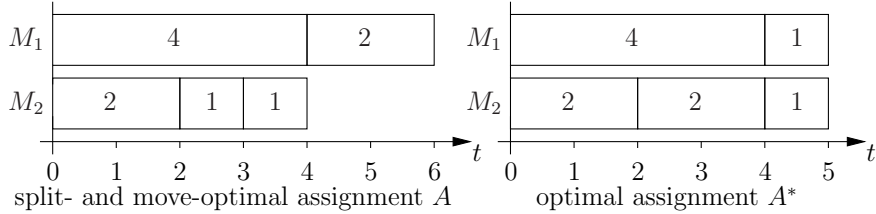


Fig. 4. Worst-case example for $m = 2$ machines.

5 Split-optimal and lexicographic-move-optimal assignments

In the previous section, we have seen that the performance guarantee of a move-optimal and split-optimal assignment marginally improves on the performance guarantee of only a move-optimal or only a split-optimal assignment. Moreover, the example, showing the tightness of the guarantee for an odd num-

ber of machines, is not lexicographic-move-optimal. Therefore, in this section we consider the lexmove-optimal and split-optimal assignments.

For lexmove-optimal assignments, we have the following fact.

Fact 10 *Let A be an assignment of the jobs to the machines and let l be a machine with minimal workload. A schedule represented by A is lexmove-optimal if and only if, for all machines i and all jobs $j \in M_i$ it holds*

$$L_l + p_j \geq L_i.$$

In this section, we only consider the LPT-algorithm as the split-operator. Remember that the LPT-algorithm sorts the jobs in non-increasing size and then iteratively assigns a job to the set with minimal workload. In this way, we obtain a partition $\text{LPT}(M_i) = (M_{i1}, M_{i2})$ that is move-optimal. Therefore, we can apply Lemma 4–8.

In the remainder of this section, we again assume w.l.o.g. that for a lexmove-optimal and split-optimal assignment A , $C_{\max}^A = 1$ and that (2) holds. Moreover, we also classify the machines into the sets $S_{<}$, S_{multi} , and S_{single} as in (5).

Lemma 11 *Let A be a lexmove-optimal and split-optimal assignment for a move-optimal split-operator. Let k be a critical machine and l be a machine with minimal load. Moreover, let C_{\max}^* denote the optimal makespan. Then, if $l \in S_{<} \cup S_{\text{multi}}$,*

$$\frac{C_{\max}^A}{C_{\max}^*} \leq \frac{3m}{2m+1}.$$

PROOF. By Lemmas 4 and 5, we can restrict ourselves to the case that $\sum_j p_j \geq mL_{k1} + L_{k2}$ and $1/2 \leq L_{k1} \leq 2/3$.

If $l \in S_{<}$, we know from Lemma 6 that $L_l \geq 2(1 - L_{k1}) \geq 2/3$ and if $l \in S_{\text{multi}}$, it follows from Lemma 6 that $L_l \geq \frac{3}{2}L_{k1} \geq 3/4 \geq 2/3$. Hence, from $L_i \geq L_l$, we have

$$C_{\max}^* \geq \frac{1}{m} \left(1 + \frac{2(m-1)}{3} \right) = \frac{2m+1}{3m}.$$

□

By this lemma, we know that in order to prove the performance guarantee of $3/2$ in Theorem 13, we can restrict ourselves to local optimal schedules with $l \in S_{\text{single}}$. Moreover, as $L_l \geq 2/3$ implies that $C_{\max}^* \geq 2/3$, we assume from here on that $L_l < 2/3$.

In the proof of Theorem 13, we use the concept of *blocking jobs*.

Definition 12 We call a job j a blocking job, if $p_j + L_{k1} \geq 2/3$, where L_{k1} is the load of the left part of a critical machine.

Note that if, in some schedule, a blocking job is assigned to the same machine as a job of size at least L_{k1} , then the makespan of this schedule will be at least $2/3$.

Theorem 13 Let A be a lexicmove-optimal and split-optimal assignment using the LPT-algorithm as split-operator. Then,

$$C_{\max}^A \leq \frac{3}{2}C_{\max}^*$$

where, C_{\max}^* denotes the optimal makespan.

PROOF. Let A be a lexicmove-optimal and split-optimal assignment and let k be a critical machine and l a machine with minimal load. By Theorem 1 we may assume w.l.o.g. that $|M_k| = 2$. Moreover, by Lemma 4 and 5, we restrict ourselves to the case that $\sum_j p_j \geq mL_{k1} + L_{k2}$ and $L_{k1} \leq 2/3$. Finally, we define the sets $S_{<}$, S_{multi} , and S_{single} as in (5). Then, by Lemma 11 we assume that $l \in S_{\text{single}}$ and $L_l < 2/3$.

Under these assumptions, we claim that for a machine $i \in S_{<} \cup S_{\text{multi}}$ the sum of processing times of blocking jobs, scheduled on this machines is at least $2/3$. None of the blocking jobs, which A assigns to a machine in $S_{<} \cup S_{\text{multi}}$, can be scheduled together with a job of size at least L_{k1} in a schedule with makespan smaller than $2/3$. Thus, in such a schedule all these jobs need to be distributed over $|S_{<} \cup S_{\text{multi}}|$ machines, as each machine in $S_{\text{single}} \cup \{k\}$ processes at least one job with processing time at least L_{k1} . From our claim, it now follows that the machine with maximal workload from these blocking jobs has workload of at least $2/3$. Hence, we always have $C_{\max}^* \geq 2/3$, and the theorem is proven.

To prove our claim, first consider a machine $i \in S_{<}$. From Lemma 6 and lexicmove-optimality of A , it follows that a job $j \in M_i$ has processing time $p_j \geq L_i - L_l \geq 4/3 - 2L_{k1}$. Hence, $p_j + L_{k1} \geq 4/3 - L_{k1} \geq 2/3$, as $L_{k1} \leq 2/3$, and j is a blocking job. As each job $j \in M_i$ is a blocking job, the total load of blocking jobs scheduled on machine $i \in S_{<}$ is $L_i \geq 2(1 - L_{k1}) \geq 2/3$.

Now, consider a machine $i \in S_{\text{multi}}$, with $|M_{i1}| \geq 3$. The smallest job in the left part, say $j_0 \in M_{i1}$ has length at most $p_{j_0} \leq L_{i1}/|M_{i1}|$. By move-optimality of the split-operator, we know that the load of the right part can be bounded by

$$L_{i2} \geq L_{i1} - p_{j_0} \geq \frac{|M_{i1}| - 1}{|M_{i1}|} L_{i1} \geq \frac{2}{3} L_{i1}.$$

Hence, by lexicmove-optimality of the assignment, we know that any job $j \in M_i$ has processing time $p_j \geq L_i - L_l \geq \frac{5}{3}L_{i1} - 2/3$. Thus $p_j + L_{k1} \geq \frac{8}{3}L_{k1} - 2/3 \geq 2/3$. Hence, each job $j \in \{i \in S_{\text{multi}} : |M_{i1}| \geq 3\}$ is a blocking job, and the total processing times of the blocking jobs assigned to such a machine i is $L_i \geq 2/3$.

Finally, consider a machine $i \in S_{\text{multi}}$, with $|M_{i1}| = 2$, say $M_{i1} = \{j_1, j_2\}$ with $p_{j_1} \geq p_{j_2}$. By move-optimality of the split-operator, we know that $L_{i2} \geq p_{j_1}$, and by lexicmove-optimality of the assignment A , we also know that $L_i \geq L_i - p_{j_2} = L_{i2} + p_{j_1} \geq 2p_{j_1}$. Hence, $p_{j_1} \leq L_i/2 \leq 1/3$.

This implies that $p_{j_2} = L_{i1} - p_{j_1} \geq L_{k1} - 1/3 \geq 1/6$ and p_{j_2} is a blocking job, as $L_{k1} + 1/6 \geq 2/3$. Moreover, due to the fact that the LPT-algorithm is used as a split-operator, we know that there exists at least one job $j \in M_{i2}$ in the right part of machine i with $p_j \geq p_{j_2}$. Hence, M_i contains at least three blocking jobs, j_1, j_2 , and j_3 , and the total processing time of these three jobs is at least

$$p_{j_1} + p_{j_2} + p_{j_3} \geq L_{k1} + 1/6 \geq 2/3,$$

which completes the proof. \square

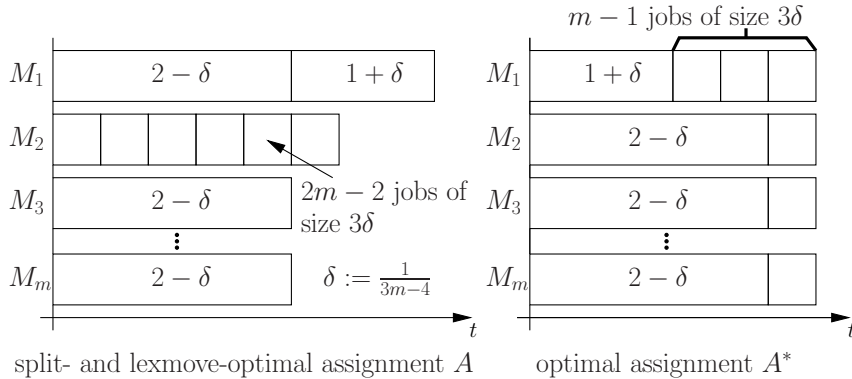


Fig. 5. A lexicmove- and split-optimal schedule A

To show a lower bound on the performance guarantee, let $\delta = \frac{1}{3m-4}$ and consider the instance consisting of $2m - 2$ jobs with processing time 3δ , one job of size $1 + \delta$ and $m - 1$ jobs of length $2 - \delta$. The assignment A as depicted in Figure 5 is lexicmove-optimal and split-optimal and has makespan $C_{\max}^A = 3$, whereas the optimal makespan is $C_{\max}^* = 2 + 2\delta$. This yields a ratio of

$$\frac{C_{\max}^A}{C_{\max}^*} = \frac{3m - 4}{2m - 2} = \frac{3}{2} - \frac{1}{2m - 2}.$$

References

- [1] E.H.L. Aarts and J.K. Lenstra (eds.), *Local search in combinatorial optimization*, John Wiley & Sons, Chichester, UK, 1997.
- [2] R.K. Ahuja, E. Özlem, J.B. Orlin, and A.P. Punnen, *A survey of very large-scale neighborhood search techniques*, Discrete Applied Mathematics **123** (2002), 75–102.
- [3] P. Brucker, J. Hurink, and F. Werner, *Improving local search heuristics for some scheduling problems II*, Discrete Applied Mathematics **72** (1997), 47–69.
- [4] T. Brueggemann, J.L. Hurink, and W. Kern, *Quality of move-optimal schedules for minimizing total weighted completion time*, Operations Research Letters (2006), to appear.
- [5] V.G. Deĭneko and G.J. Woeginger, *A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem*, Mathematical Programming, Series A **87** (2000), 519–542.
- [6] G. Finn and E. Horowitz, *A linear time approximation algorithm for multiprocessor scheduling*, BIT **19** (1979), 312–320.
- [7] M.R. Garey and D.S. Johnson, *Complexity results for multiprocessor scheduling under resource constraints*, SIAM Journal on Computing **4** (1975), 397–411.
- [8] 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*, Annals of Discrete Mathematics **5** (1979), 287–326.
- [9] R.L. Graham, *Bounds on multiprocessing anomalies*, SIAM Journal on Applied Mathematics **17** (1969), 416–429.
- [10] C.A.J. Hurkens and T. Vredeveld, *Local search for multiprocessor scheduling: How many moves does it take to a local optimum?*, Operations Research Letters **31** (2003), 137–141.
- [11] P. Schuurman and T. Vredeveld, *Performance guarantees of local search for multiprocessor scheduling*, Inform Journal on Computing (2006), to appear.