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

Tobias Brueggemann[1], Johann L. Hurink[1], Tjark Vredeveld[2], and Gerhard J. Woeginger[3]

[1] Dept. of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
j.l.hurink@utwente.nl
[2] Dept. of Quantitative Economics, Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands
t.vredeveld@ke.unimaas.nl
[3] Dept. of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
gwoegi@win.tue.nl

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 for every machine the set of assigned jobs into two sets by some fixed rule and then to reassign these $2m$ parts such that every machine gets exactly two parts. The split neighborhood consists of all possible reassignments of the parts and a best neighbor can be calculated in $\mathcal{O}(m \log m)$ by determining a perfect matching with minimum maximal edge weight.

We examine local optima in the split neighborhood and in combined neighborhoods consisting of the split and other known neighborhoods and derive performance guarantees for these local optima.

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 [9], this problem is denoted as $P||C_{\max}$.

This problem is known to be strongly NP-hard for m being part of the input [8]. Therefore, we search for approximate solutions. If an algorithm is guaranteed

* Supported by the Netherlands Organization for Scientific Research (NWO) grant 613.000.225 (Local Search with Exponential Neighborhoods) and by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society).

C. Kaklamanis and M. Skutella (Eds.): WAOA 2007, LNCS 4927, pp. 41–54, 2008.

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 [10]: 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 improvement. For the parallel machine scheduling problem under consideration, Finn and Horowitz [7] 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 [12]. Brucker et al. [3] showed that the iterative improvement procedures needs $\mathcal{O}(n^2)$ moves to come to a local optimal solution, and this bound is tight [11]. For performance guarantees of local search methods regarding makespan minimization, we refer to [12,11]. For the objective of minimizing total weighted completion time, Brueggemann et al. [4] give 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 [6] 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, the so-called *split-neighborhood*, and in the following sections we investigate its worst-case behavior. In Section 3, we show that a split-optimal solution has the same performance guarantee as a simple move-optimal solution. 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 show 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 the used 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, $A : J \to \{1, \ldots, m\}$, where $J = \{1, \ldots, n\}$ denotes the set of jobs. Each assignment leads to a partition of the set of jobs into m disjoint subsets $M_1^A, \ldots, M_m^A$, where $M_i^A = \{j \in J : A(j) = i\}$ is the set of jobs scheduled on machine i. Abusing terminology, we use "schedule A" for any schedule complying to the assignment A. 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. Given a schedule A, we select a job j, scheduled on machine h, and a machine $i \neq h$. The move neighbor, A', is obtained by moving job j to machine i, i.e., $M_h^{A'} = M^A \setminus \{j\}$, $M_i^{A'} = M^A \cup \{j\}$, and $M_k^{A'} = M^A$ for $k \neq h, i$. The set of all move neighbors of schedule A is denoted by $\mathcal{N}_{\text{move}}(A) = \{A' : A' \text{ is a move neighbor of } A\}$. We call an assignment A *move-optimal* if for all move neighbors 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 [7] gave the following upper bound on the performance guarantee of move-optimal assignments.

Theorem 1 ([7]). *Let A be a move-optimal assignment, and let $C_{\max}^*$ denote the optimal makespan. Moreover, let $n_k = \max\{\, |M_i| : L_i = C_{\max}^A \,\}$ denote 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 [12], see Figure 1 for the assignments attaining this bound.

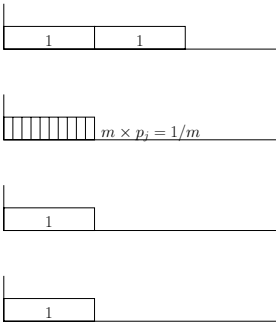

Fig. 1. Worst-case move-optimal schedule

The split-neighborhood. The *split-neighborhood* is of exponential size in the number of machines. The basis of this neighborhood is a *split-operator* that partitions the set of jobs assigned to a machine i into two disjoint sets, i.e., $\mathsf{split}(M_i) = (M_{i1}, M_{i2})$, where M_i denotes the set of jobs scheduled on machine i and (M_{i1}, M_{i2}) is a partition of the set M_i into two disjoint subsets. We assume w.l.o.g. that $L_{i1} = \sum_{j \in M_{i1}} p_j \geq \sum_{j \in M_{i2}} p_j = L_{i2}$. We refer to the sets M_{i1} and M_{i2} of a machine i (or a set M_i) as the *left* 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_{i_1} and M_{i_2} for $i = 1, \ldots, m$. Abusing notation, we denote the set of these $2m$ parts by

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

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

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

Although the size of the neighborhood is exponentially large in the number of machines, the following fact tells us that the best neighbor, one with lowest makespan and fewest number of critical machines among all neighbors with lowest makespan, can be found in $\mathcal{O}(m \log m)$ time.

Fact 1. *Given $2m$ numbers $a_1 \geq \ldots \geq a_{2m}$. A perfect matching of these numbers such that the maximum of the sum of two matched numbers is minimized, is obtained by matching a_i to a_{2m+1-i}. Moreover, this matching minimizes the number of matched 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 first rearranging the $2m$ parts $T_1, \ldots, T_{2m}$ in $\mathsf{split}(A)$,

so that for the sum of processing times of the parts holds $L_{T_1} \geq \ldots \geq L_{T_{2m}}$ and, then, assigning the jobs of the parts T_i and T_{2m+1-i} to machine i for $i = 1, \ldots, 2m$.

We call an assignment *split-optimal* if for all $A' \in \mathcal{N}_{\text{split}}(A)$, $C^A_{\max} \leq C^{A'}_{\max}$ and in case $C^A_{\max} = C^{A'}_{\max}$ the number of critical machines in A is at most the number of critical machines in A'. Of course, the quality of a split-optimal solution depends on the split-operator. For most of the presented results, we only assume that the split-operator produces a move-optimal partition. That is, for any job $j \in M_{i1}$, we have that $L_{i2} + p_j \geq L_{i1}$. Such a split-operator we call a move-optimal split-operator.

Combinations of move and split-neighborhood. As we will see in the following section, a split-optimal assignment needs not to be move-optimal. Hence, we also consider assignments that are both move- and split-optimal. These local optima may however be improved by first applying a move-operator leading to a schedule with the same makespan and, then applying a split-operator leading to a better schedule. Therefore, we define a *lexicographic-move-optimal* assignment. For a given assignment A and $A' \in \mathcal{N}_{\text{move}}(A)$, we reorder the machines in A and A' so that

$$L^A_1 \geq \ldots \geq L^A_m \quad \text{and}$$
$$L^{A'}_1 \geq \ldots \geq L^{A'}_m.$$

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

$$\begin{aligned} L^{A'}_i &= L^A_i \quad \text{for } i = 1, \ldots, k-1, \\ L^{A'}_k &< L^A_k. \end{aligned} \tag{1}$$

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

Recall that we assume that a split-operator on a set M_i produces two parts M_{i1} and M_{i2} satisfying $L_{i1} \geq L_{i2}$. Moreover, for a critical machine k in a split-optimal schedule, we have the following property 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 2. *Let A be a split-optimal assignment using a move-optimal split-operator. Then the makespan of A is bounded by $C^A_{\max} \leq (2 - \frac{2}{m+1})C^*_{\max}$, where $C^*_{\max}$ denotes the value of the optimal makespan.*

Proof. W.l.o.g. we assume that $C^A_{\max} = 1$. 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

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

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_{k1} and M_{k2}, we know that for all $j \in M_{k1}$, $L_{k2} + p_j \geq L_{k1}$. Therefore, from (4) it follows that $p_j > \frac{1}{2}L_{k1}$ for all $j \in M_{k1}$. Hence, M_{k1} contains only one job and

$$C^A_{\max} = L_k = L_{k1} + L_{k2} < \frac{3}{2}L_{k1} \leq \frac{3}{2}C^*_{\max},$$

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 $\frac{3}{2} \leq 2 - 2/(m+1)$. 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

$$C^A_{\max} \leq \frac{4}{3}C^*_{\max}.$$

□

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 $C^A_{\max} = 2$. 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 $C^A_{\max} = 2m/(m+1)C^*_{\max}$. See Figure 2 for an illustration.

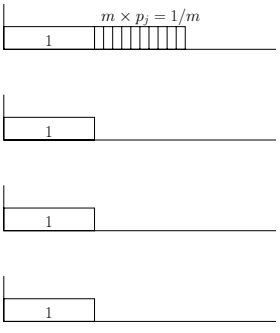

Fig. 2. A split-optimal assignment

4 Split-Optimal and Move-Optimal Assignments

The worst-case instance for split-optimal assignments, showing the tightness of the 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 3, for move-optimal split-operators.

Lemma 1. *Let A be a move-optimal and split-optimal assignment using a move-optimal split-operator. If there exists a critical machine k such that*

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

then $C^A_{\max} = C^*_{\max}$, 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, $C^A_{\max} = L_k = p_{j_1} \leq C^*_{\max}$. □

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 3.

Lemma 2. *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 $L_{k1} \geq \frac{2}{3}L_k$, then the makespan of A can be bounded by*

$$C^A_{\max} \leq \frac{3m}{2m+1} C^*_{\max},$$

where $C^*_{\max}$ denotes the optimal makespan.

Proof. By Lemma 1, we only have to consider the case that $\sum_j p_j \geq mL_{k1} + L_{k2}$. Hence, the optimal makespan can be bounded from below by $C^*_{\max} \geq \frac{1}{m}\sum_j p_j \geq \frac{(m-1)L_{k1}+L_k}{m}$. As, $C^A_{\max} = L_k$, we thus have

$$\frac{C^A_{\max}}{C^*_{\max}} \leq \frac{m}{(m-1)L_{k1} + L_k} \leq \frac{3m}{2m+1},$$

where the last inequality is due to $L_{k1} \geq 2/3L_k$. □

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

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

That is, $S_<$ is the set of machines that have a left part which is smaller than L_{k1}. This set of remaining machines 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 can be bounded as follows.

Lemma 3. *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:*

$$\begin{aligned} L_i &\geq 2(C^A_{\max} - L_{k1}) \quad &&\text{for} \quad i \in S_<, \\ L_i &\geq \tfrac{3}{2}L_{k1} \quad &&\text{for} \quad i \in S_{multi}. \end{aligned}$$

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 $L_i \geq 2L_{i2} \geq 2L_{k2} = 2(1 - L_{k1})$.

For $i \in S_{\text{multi}}$ 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, $L_i \geq \frac{3}{2}L_{i1} \geq \frac{3}{2}L_{k1}$. □

Lemma 4. *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 $L_{k1} \leq 2/3L_k$ and $|S_<| \geq 1$, then*

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

where $C^*_{\max}$ denotes the optimal makespan.

Proof. Using Lemma 3, we can bound the optimal makespan by

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

where the last inequality is due to $L_{k1} \leq 2/3L_k = 2/3C^A_{\max}$. 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 $1/2C^A_{\max} \leq L_{k1} \leq 2/3C^A_{\max}$, we have

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

□

Lemma 5. *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 $L_{k1} \leq 2/3L_k$ and $|S_{multi}| \geq 2$, then*

$$\frac{C^A_{\max}}{C^*_{\max}} \leq \frac{2m}{m+2},$$

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 $L_{k1} \leq 2/3L_k$, we know from Lemma 3 that for $i \in S_<$, $L_i \geq L_{k1}$. Hence, using Lemma 3, we can bound the optimal makespan by

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

where the second inequality is due to $|S_{\text{multi}}| \geq 2$ and the last to $L_{k1} \geq C^A_{\max}/2$. □

Theorem 3. *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. Assume that $C^A_{\max} = 1$. Since $\frac{3m}{2m+1} \leq 2 - \frac{4}{m+3}$, by Theorem 1 we only need to consider assignments A in which a critical machine k contains exactly two jobs. Moreover, by Lemma 2, we may restrict ourselves to the case that $L_{k1} \in [\frac{1}{2}, \frac{2}{3}]$, and by Lemma 1, we may assume that $\sum_j p_j \geq mL_{k1} + L_{k2}$.

As $\max\{\frac{2m}{m+2}, \frac{3m}{2m+1}\} \leq 2 - \frac{4}{m+3}$, due to Lemma 4 we can restrict ourselves to the case that there is no machine i with $L_{i1} < L_{k1}$. Due to Lemma 5 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 $C^*_{\max} = C^A_{\max}$. 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

$$\frac{C^A_{\max}}{C^*_{\max}} \leq \frac{2m+2}{m+3}.$$ □

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 $C^A_{\max} = 2$. 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 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 simular as in Figure 3.

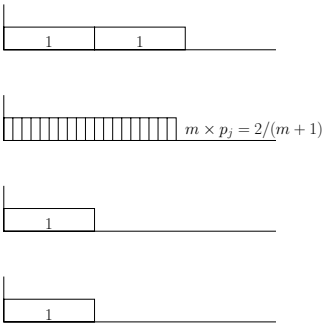

Fig. 3. A split and move-optimal assignment for odd number of machines

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 number 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 2. *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 $\mathsf{LPT}(M_i) = (M_{i1}, M_{i2})$ that is move-optimal. Therefore, we can apply Lemma 1–5.

Lemma 6. *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 $l \in S_< \cup S_{multi}$,*

$$\frac{C^A_{\max}}{C^*_{\max}} \leq \frac{3m}{2m+1}.$$

Proof. Assume w.l.o.g. that $C^A_{\max} = 1$. By Lemmas 1 and 2, we can restrict ourselves to the case that $\sum_j p_j \geq mL_{k1} + L_{k2}$ and $L_{k1} \leq 2/3$.

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

$$C^*_{\max} \geq \frac{1}{m}\big(1+\frac{2(m-1)}{3}\big) = \frac{2m+1}{3m}. \qquad \square$$

By this lemma, we know that in order to prove the performance guarantee of 3/2 in Theorem 4, we can restrict ourselves to local optimal schedules with $l \in S_{\text{single}}$. Moreover, as $L_l \geq 2/3C^A_{\max}$ implies that $C^*_{\max} \geq 2/3C^A_{\max}$, we assume in the remainder of this section that $L_l < 2/3C^A_{\max}$.

In the proof of Theorem 4, we use the concept of *blocking jobs*.

Definition 1. *We call a job j a* blocking job*, if $p_j + L_{k1} \geq 2/3C^A_{\max}$, 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 $2/3C^A_{\max}$. The idea of the proof of the following theorem is that the total volume of blocking jobs is large enough so that if no blocking job is assigned on the same machine as a job of size at least L_{k1}, then the makespan of the schedule is also at least $2/3C^A_{\max}$.

Theorem 4. *Let A be a lexmove-optimal and split-optimal assignment using the LPT-algorithm as split-operator. Then,*

$$C^A_{\max} \leq \frac{3}{2}C^*_{\max},$$

where, $C^*_{\max}$ denotes the optimal makespan.

Proof. Let A be a lexmove-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 1 and 2, we restrict ourselves to the case that $\sum_j p_j \geq mL_{k1} + L_{k2}$ and $L_{k1} \leq 2/3L_k$. Finally, we define the sets $S_<$, S_{multi}, and S_{single} as in (5). Then, by Lemma 6 we assume that $l \in S_{\text{single}}$ and $L_l < 2/3C^A_{\max}$.

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 $2/3C^A_{\max}$. 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 $2/3C^A_{\max}$. 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 in every feasible schedule the blocking jobs lead on at least one machine to a workload of at least $2/3C^A_{\max}$. Hence, we always have $C^*_{\max} \geq 2/3C^A_{\max}$, and the theorem is proven.

To prove our claim, first consider a machine $i \in S_<$. From Lemma 3 and lexmove-optimality of A, it follows that a job $j \in M_i$ has processing time $p_j \geq$

$L_i - L_l \geq 4/3C^A_{\max} - 2L_{k1}$. Hence, $p_j + L_{k1} \geq 4/3C^A_{\max} - L_{k1} \geq 2/3C^A_{\max}$, as $L_{k1} \leq 2/3L_k = 2/3C^A_{\max}$, 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 $L_i \geq 2(C^A_{\max} - L_{k1}) \geq 2/3C^A_{\max}$.

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 lexmove-optimality of the assignment, we know that any job $j \in M_i$ has processing time $p_j \geq L_i - L_l \geq \frac{5}{3}L_{i1} - 2/3C^A_{\max}$. Thus $p_j + L_{k1} \geq \frac{8}{3}L_{k1} - 2/3C^A_{\max} \geq 2/3C^A_{\max}$. 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 $L_i \geq 2/3C^A_{\max}$.

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 lexmove-optimality of the assignment A, we also know that $L_l \geq L_i - p_{j_2} = L_{i2} + p_{j_1} \geq 2p_{j_1}$. Hence, $p_{j_1} \leq L_l/2 \leq 1/3C^A_{\max}$.

This implies that $p_{j_2} = L_{i1} - p_{j_1} \geq L_{k1} - 1/3C^A_{\max} \geq 1/6C^A_{\max}$ and p_{j_2} is a blocking job, as $L_{k1} + 1/6C^A_{\max} \geq 2/3C^A_{\max}$. 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

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

which completes the proof. □

A lower bound on the performance guarantee is given in Figure 4. Let $\delta = \frac{1}{3m-4}$ and consider the instance consisting of $2m-2$ jobs with processing time 3δ, one

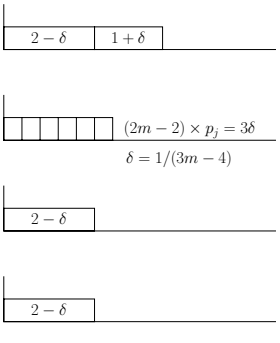

Fig. 4. A lexmove- and split-optimal schedule A

job of size $1+\delta$ and $m-1$ jobs of length $2-\delta$. The assignment A as depicted in Figure 4 is lexmove-optimal and split-optimal and has makespan $C^A_{\max} = 3$, whereas the optimal makespan is $C^*_{\max} = 2+2\delta$. This yields a ratio of The ratio between the lexmove- and split-optimal schedule depicted in this figure and the optimal makespan is $\frac{C^A_{\max}}{C^*_{\max}} = \frac{3m-4}{2m-2} = \frac{3}{2} - \frac{1}{2m-2}$.

References

1. Aarts, E.H.L., Lenstra, J.K. (eds.): Local search in combinatorial optimization. John Wiley & Sons, Chichester (1997)
2. Ahuja, R.K., Özlem, E., Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics 123, 75–102 (2002)
3. Brucker, P., Hurink, J.L., Werner, F.: Improving local search heuristics for some scheduling problems II. Discrete Applied Mathematics 72, 47–69 (1997)
4. Brueggemann, T., Hurink, J.L., Kern, W.: Quality of move-optimal schedules for minimizing total weighted completion time. Operations Research Letters 34, 583–590 (2006)
5. Brueggemann, T., Hurink, J.L., Vredeveld, T., Woeginger, G.J.: Very large-scale neighborhoods with performance guarantees for minimizing makespan on parallel machines, Tech. Report No. 1801, University of Twente, Dep. of Mathematical Sciences (2006)
6. Deĭneko, V.G. Woeginger, G.J.: A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. Mathematical Programming, Series A 87, 519–542 (2000)
7. Finn, G., Horowitz, E.: A linear time approximation algorithm for multiprocessor scheduling. BIT 19, 312–320 (1979)
8. Garey, M.R., Johnson, D.S.: Complexity results for multiprocessor scheduling under resource constraints. SIAM Journal on Computing 4, 397–411 (1975)
9. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5, 287–326 (1979)
10. Graham, R.L.: Bounds on multiprocessing anomalies. SIAM Journal on Applied Mathematics 17, 416–429 (1969)
11. Hurkens, C.A.J., Vredeveld, T.: Local search for multiprocessor scheduling: How many moves does it take to a local optimum? Operations Research Letters 31, 137–141 (2003)
12. Schuurman, P., Vredeveld, T.: Performance guarantees of local search for multiprocessor scheduling. Informs Journal on Computing 19, 52–63 (2007)