Performance Ratios for the Karmarkar-Karp Differencing Method

Wil Michiels[1,2], Jan Korst[2], Emile Aarts[1,2], and Jan van Leeuwen[3]

[1] Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[2] Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
[3] Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
michiels@natlab.research.philips.com

Abstract. We consider the multiprocessor scheduling problem in which one must schedule n independent tasks nonpreemptively on m identical, parallel machines, such that the completion time of the last task is minimal. For this well-studied problem the Largest Differencing Method due to Karmarkar and Karp outperforms other existing polynomial-time approximation algorithms from an average-case perspective. For $m \geq 3$, its worst-case performance has remained a challenging open problem. In this paper, we show that its performance ratio is bounded between $\frac{4}{3} - \frac{1}{3(m-1)}$ and $\frac{4}{3} - \frac{1}{3m}$. We also analyze the performance ratio if in addition to the number of machines, the number of tasks n is fixed as well.

1 Introduction

A classical problem in complexity theory is the multiprocessor scheduling problem in which we have to find a nonpreemptive schedule of n independent tasks on m identical, parallel machines, such that the completion time of the last task is minimal. Using the three-field notation introduced by Graham et al. [10] the problem can be written as $P||C_{\max}$. In the literature, multiprocessor scheduling is also called number partitioning as it can be viewed as the problem of partitioning a set of n numbers into m subsets, such that the maximum subset sum is minimal.

For this strongly NP-hard problem [7], the Largest Differencing Method (LDM) of Karmarkar and Karp [13] outperforms other polynomial-time approximation algorithms such as Longest Processing Time (LPT) [9] and Multifit [2] from an average-case perspective [3, 16, 21]. Surprisingly enough, its worst-case performance has remained open for several years. Although for the special case that $m = 2$, Fischetti and Martello [4] already showed that LDM has a performance ratio of $\frac{7}{6}$, the worst-case performance of LDM was still unknown for $m \geq 3$. This issue is settled in this paper.

Definition (Multiprocessor Scheduling/ Number Partitioning). A problem instance I is defined by an integer m and a set $A = \{1, 2, \ldots, n\}$ of n items, where each item $j \in A$ has a nonnegative size a_j with $a_1 \leq a_2 \leq \cdots \leq a_n$. Find a partition $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ of A into m disjoint subsets, such that

$$f_I(\mathcal{A}) = \max_{1 \leq i \leq m} S(A_i)$$

is minimal, where $S(A_i) = \sum_{j \in A_i} a_j$. □

We first explain LDM by means of an example. Consider the problem instance with $m = 2$ and items with sizes $4, 5, 6, 7$, and 8. In the first iteration, LDM selects the two largest item sizes 8 and 7 and commits both item sizes to different subsets. The decision to which subsets they are actually assigned is equivalent to deciding to which subset we assign their absolute difference, i.e., $8 - 7 = 1$. Therefore, LDM replaces the items with size 8 and 7 by a new item with size 1. This operation is called differencing. The described process is now repeated until a single item remains; see Figure 1. This means

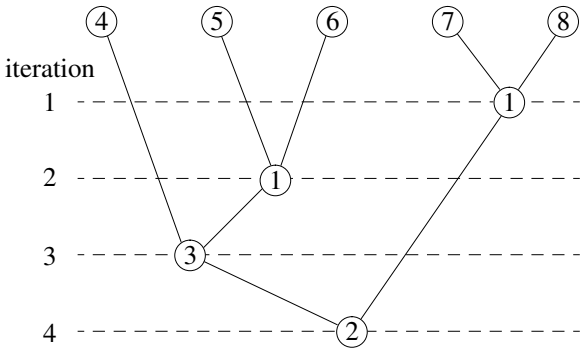

Figure 1. Visualization of the successive iterations of LDM.

that in the second iteration, LDM replaces the items with size 6 and 5 by an item with size 1, which leaves the three item sizes $4, 1$, and 1. In the third and fourth iteration, the algorithm replaces 4 and 1 by 3 and 3 and 1 by 2, respectively. By backtracing through the successive differencing operations, we can easily determine the resulting subsets $A_1 = \{7, 5, 4\}$ and $A_2 = \{8, 6\}$ that partition the items and for which the difference in sum equals the size of the last remaining item in the sequence. For clarity, we defined the subsets A_1 and A_2 by giving their item sizes instead of their items. In Section 2 we give a more precise discussion of the algorithm for $m \geq 2$.

In this paper we say that an algorithm has a performance bound U if it always delivers a solution with a cost at most U times the optimal cost. If bound U is tight, then U is called a performance ratio.

Related work. For $m = 2$, Yakir [23] proves that if the item sizes are uniformly distributed over $[0, 1]$, then the expected difference between the sum of the two subsets in a partition generated by LDM is $n^{-\Theta(\log n)}$. This implies that also the expected deviation of the cost of such a partition from the optimal cost is $n^{-\Theta(\log n)}$. Also for $m = 2$, the average-case performance of alternative implementations of the differencing method are studied in [1, 15, 22]. The implementations differ from LDM in the choice of the items that are selected for differencing.

For given $m \geq 2$, Karmarkar and Karp [13] present a rather elaborate differencing method. The algorithm uses some randomization in selecting the pair that is to be differenced so as to facilitate its probabilistic analysis. For the algorithm, they prove that the difference between the maximum and minimum subset sum is at most $n^{-O(\log n)}$,

almost surely, when the item sizes are in $[0,1]$ and the density function is reasonably smooth.

Korf [14] proposes a branch-and-bound algorithm that starts with LDM and then tries to find a better solution until it ultimately finds and proves the optimal solution. Although the algorithm is practically useful for $m = 2$, it is less interesting for $m \geq 3$. Other extensions of the differencing method are given by Ruml et al. [19] and Storer et al. [20], who present successful local search heuristics based on the differencing method.

Balanced number partitioning is defined as multiprocessor scheduling with the additional constraint that the cardinality of the subsets is balanced, which means that each subset contains either $k = \lceil n/m \rceil$ or $\lfloor n/m \rfloor$ items. For this problem Yakir [23] proposes the Balanced Largest Differencing Method (BLDM). For $m = 2$ the algorithm works as follows. First, it starts by differencing the largest two items, the third and fourth largest item, etc. and next it proceeds with LDM. As mentioned, LDM has a better average-case performance than any other known polynomial-time algorithm for multiprocessor scheduling. The same can be said for BLDM with respect to balanced number partitioning. Michiels, Korst, Aarts, and Van Leeuwen [18] determine the worst-case performance of BLDM both as a function of m and as a function of k. Furthermore, for $m = 2$ Yakir [23] shows that as for LDM, the expected difference between the sum of the two subsets in a partition generated by BLDM is $n^{-\Theta(\log n)}$ for some constant c if the item sizes are uniformly distributed over $[0,1]$

Hochbaum and Shmoys [12] show that Multiprocessor scheduling allows a PTAS [12]. By choosing a sufficiently small precision ε, the PTAS yields a polynomial-time algorithm for which its worst-case performance as well as its average-case performance will outperform the differencing method. However, as the running time grows exponentially in $\frac{1}{\varepsilon}$, the algorithm will not be practically useful. Hence, our claim that the differencing method is the best polynomial-time algorithm from an average-case perspective has to be specified to that it is the best practical polynomial-time algorithm. Therefore, it is interesting to determine its worst-case performance.

Several polynomial-time approximation algorithms exist that are competitive to LDM. Of those, the two most popular are LPT [9] and Multifit [2]. In the former algorithm the items are assigned in decreasing order to the subset with minimum sum. Graham [9] proves that the algorithm has a performance ratio of $\frac{4}{3} - \frac{1}{3m}$. In addition, Frenk and Rinnooy Kan [5] show that if the item sizes are uniformly distributed over $[0,1]$, then the difference between the cost of a partition given by LPT and the optimal cost is at most $O(\log n/n)$ almost surely and $O(1/n)$ in expectation. Note that this is worse than the average case results for LDM.

Multifit, on the other hand, performs a binary search to find the minimum bin-capacity for which the bin-packing algorithm First Fit Decreasing (FFD) finds a feasible solution, where FFD assigns the items in decreasing order to the first bin in which they fit. Coffman, Garey, and Johnson [2] prove that the performance ratio of Multifit is $\frac{8}{7}$ for $m = 2$, $\frac{15}{13}$ for $m = 3$, and $\frac{20}{17}$ for $m = 4,5,6$, and 7. For $m \geq 8$, Yue [24] derives a performance bound of $\frac{13}{11}$ and Friesen [6] shows that this bound is tight for any $m \geq 13$.

Finally, we mention recent results on the multiprocessor scheduling problem that have been inspired by statistical physics. Mertens [17] analyzes the phase boundary

that separates easy problem instances from hard ones for $m = 2$. The results support the paper of Gent and Walsh [8], who already gave computational evidence for the existence of a phase transition. A nice introductory overview on the issue is given by Hayes [11].

Our results. In Section 2 we discuss LDM for the general case $m \geq 2$. To provide evidence for the good average-case performance of LDM, we also compare some simulation results of LDM, LPT, and Multifit in Section 2.

In Section 3, we prove that the worst-case performance ratio of LDM is bounded between $\frac{4}{3} - \frac{1}{3(m-1)}$ and $\frac{4}{3} - \frac{1}{3m}$. This implies that opposite to its superior average-case performance, LDM has a worst-case performance that is worse than Multifit, but at least as good as LPT. We stress that the proof of our results cannot be based on the derivations given by Graham [9], who proves the $\frac{4}{3} - \frac{1}{3m}$ performance bound for LPT, nor on those by Fischetti and Martello [4], who analyze the worst-case performance of LDM for $m = 2$.

An interesting question is whether these performance results improve if we have additional information on n, for instance if we are given that we only apply LDM on problem instances for which the average number of items per subset is small. Section 4 discusses the performance ratio of LDM as a function of both m and n. For $m = 2$, the results of Fischetti and Martello [4] imply that LDM is optimal if $n \leq 4$ and that it has a performance ratio of $\frac{7}{6}$, otherwise. For $m \geq 3$, we derive that LDM is optimal for $n \leq m+2$ and that it has a performance ratio of $\frac{4}{3} - \frac{1}{2(n-m-1)}$ for $m+2 \leq n \leq 2m$. For given $n > 2m$, we prove that the performance ratio is again bounded by $\frac{4}{3} - \frac{1}{3(m-1)}$ and $\frac{4}{3} - \frac{1}{3m}$.

2 Largest Differencing Method

We first introduce some notational conventions. Next, we discuss LDM for any $m \geq 2$ and we present some simulation results to show its good average-case performance.

When we add a star as superscript to our notation, this indicates optimality. For example, f_I^* gives the objective value of an optimal partition $\mathcal{A}^*$ for a problem instance I. Furthermore, we represent sets of items by giving a sequence of their sizes. This means that instead of $A = \{1,2,3\}$ with $a_i = 5+i$ we write $(6,7,8)$ or $A = 6,7,8$, for short. Moreover, A_i - A_{i+1} - ... - A_m denotes the partition $\{A_1, A_2, \ldots, A_m\}$ in which $A_1, A_2, \ldots, A_{i-1}$ are empty and A_i^l is a short-hand notation for A_j - A_j - ... - A_j (l times).

Initially, LDM [13] starts with a sequence L of the n partial solutions $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$, where each subset in $\mathcal{A}_i = \{A_{i1}, A_{i2}, \ldots, A_{im}\}$ is empty except for $A_{im} = \{i\}$. Next, the algorithm executes $n-1$ iterations. In each iteration, it selects the two partial solutions from L for which $d(\mathcal{A})$ is maximal, where $d(\mathcal{A})$ is defined as the difference between the maximum and minimum subset sum in $\mathcal{A}$. These two solutions, denoted by $\mathcal{A}'$ and $\mathcal{A}''$, are combined into a new partial solution $\mathcal{A}$ by joining the subset with smallest sum in $\mathcal{A}'$ with the subset with largest sum in $\mathcal{A}''$, the subset with second smallest sum in $\mathcal{A}'$ with the subset with second largest sum in $\mathcal{A}''$, and so on. Hence, $\mathcal{A}$ is formed by the m subsets $A'_j \cup A''_{m-j+1}$ for $1 \leq j \leq m$, where the subsets of $\mathcal{A}'$ and $\mathcal{A}''$ have been put in order of non-decreasing sum. Solution $\mathcal{A}$ replaces $\mathcal{A}'$ and $\mathcal{A}''$ in L. After $n-1$ of

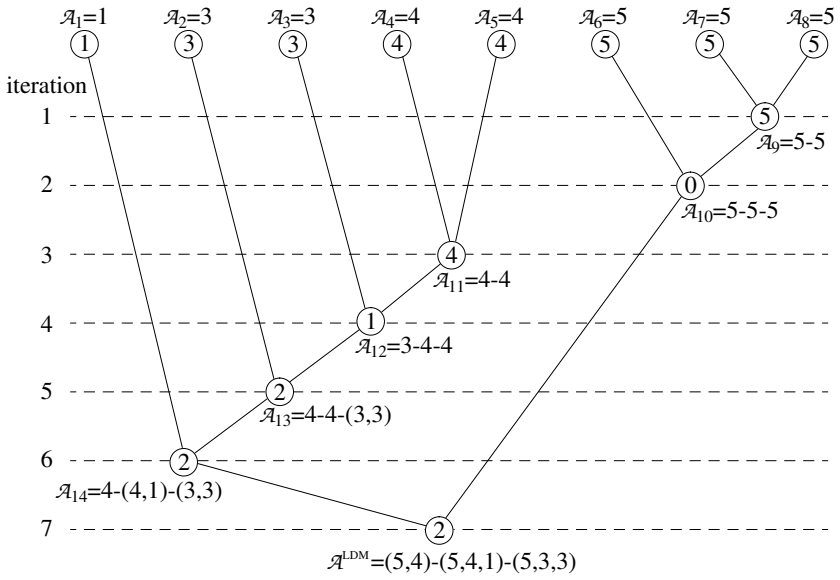

Figure 2. Visualization of the successive iterations of LDM. A circle represents a partial solution $\mathcal{A}$ and the number inside the circle equals $d(\mathcal{A})$.

such so-called differencing operations, only one solution in L remains. This solution is called $\mathcal{A}^{\text{LDM}}$. We illustrate the algorithm, which has a time complexity of $O(n \log n)$, by means of an example.

Example 1. Let $n = 8$, $m = 3$, and the eight item sizes be 1,3,3,4,4,5,5,5. Initially, L consists of the partial solutions $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_8$, where $d(\mathcal{A}_i)$ equals the size of the only item in the ith partition, i.e., $d(\mathcal{A}_i) = a_i$. In the first iteration, $\mathcal{A}_7$ and $\mathcal{A}_8$ are replaced by $\mathcal{A}_9 = 5$ - 5 with $d(\mathcal{A}_9) = 5$; see Figure 2. Next, the algorithm differences $\mathcal{A}_6$ and $\mathcal{A}_9$. This yields partial solution $\mathcal{A}_{10} = 5$ - 5 - 5 with $d(\mathcal{A}_{10}) = 0$. After five more iterations, which are depicted in Figure 2, we obtain $\mathcal{A}^{\text{LDM}} = (4,5)$ - $(1,4,5)$ - $(3,3,5)$ for which the maximum subset sum is 11. This partition is not optimal as the maximum subset sum of optimal partition $\mathcal{A}^* = (5,5)$ - $(3,3,4)$ - $(1,4,5)$ is 10. □

Next, we present a number of simulation results of LDM, LPT, and Multifit. Thereby, it is not our goal to give an elaborate study on the average-case performance of known polynomial time algorithms for multiprocessor scheduling, but to indicate the good average-case performance of LDM by comparing its simulation results with those of the two most popular algorithms for multiprocessor scheduling.

In our first experiment, we study the performance of the three algorithms for $m = 10$ and for n ranging from 1 to 250. For each n, we generated 10,000 problem instances with item sizes uniformly distributed over $[0,1]$. Figure 3 depicts the average deviation of the cost from the obvious lower bound $\max(a_n, S(A)/m)$. We see that LDM outper-

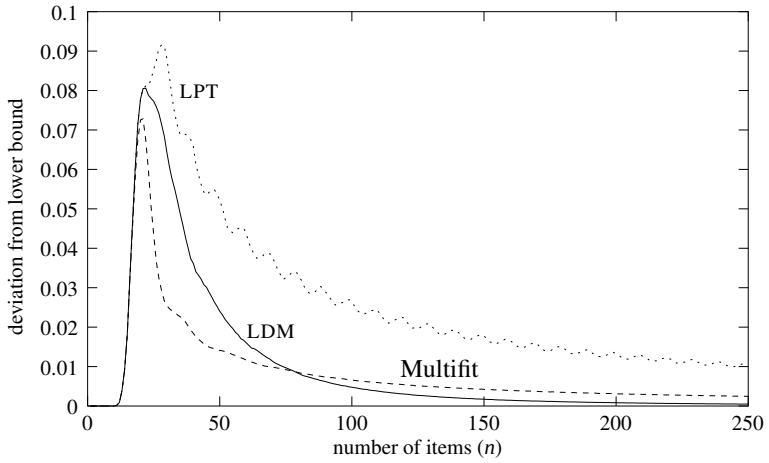

Figure 3. Average deviation of the cost of partitions given by LDM, LPT, and Multifit from the lower bound $\max(a_n, S(A)/m)$, where $m = 10$ and the item sizes are uniformly distributed over $[0, 1]$. For each n, we generated 10,000 problem instances.

forms LPT for all n and that LDM outperforms Multifit from $n = 79$ in which case on average 8 items are assigned to each subset. The same trend can be observed for other m. However, the turnover point from which LDM outperforms Multifit increases with the average number of items per subset. For $m = 2$ it is 3, whereas for $m = 20$ it is 14.

In our second experiment, we consider a positive offset o. This means that we let the item sizes be uniformly distributed over $[o, o+1]$ instead of on $[0, 1]$ for some offset o. This corresponds to assuming that $\frac{a_n}{a_1}$ is bounded by $1 + \frac{1}{o}$. In our experiment, we let o range from 0 to 1 and we let $m = 10$ and $n = 100$. The results are given in Figure 4. The performance of Multifit strongly deteriorates for increasing o. This can be explained as follows. Each iteration of Multifit consist of an execution of the algorithm FFD for a given bin capacity. If during the execution of FFD we get a bin in which less than o is unused, then no items will be assigned to that bin in the remainder of the execution of FFD as all item sizes are larger than o. Hence, for large o, we have a high probability that FFD leaves much of the bin capacity unused, which implies a poor performance.

Also for LPT we have a performance that is worse than for LDM. Thereby, we note that m divides n in this experiment. It can be argued that if this is not the case, then the performance of LPT gets worse, especially for a small value of n mod m.

3 Performance ratio as a function of m

In the remainder of this paper, we study the worst-case performance of LDM. First, we analyze the performance ratio as a function of $m \geq 2$. Although Fischetti and Martello [4] already prove a performance ratio of $\frac{7}{6}$ for $m = 2$, we nevertheless consider the case $m = 2$ as this result is also implied by our analysis. For $m \geq 3$, we show that the performance ratio is bounded between $\frac{4}{3} - \frac{1}{3m}$ and $\frac{4}{3} - \frac{1}{3(m-1)}$. However, we first derive some

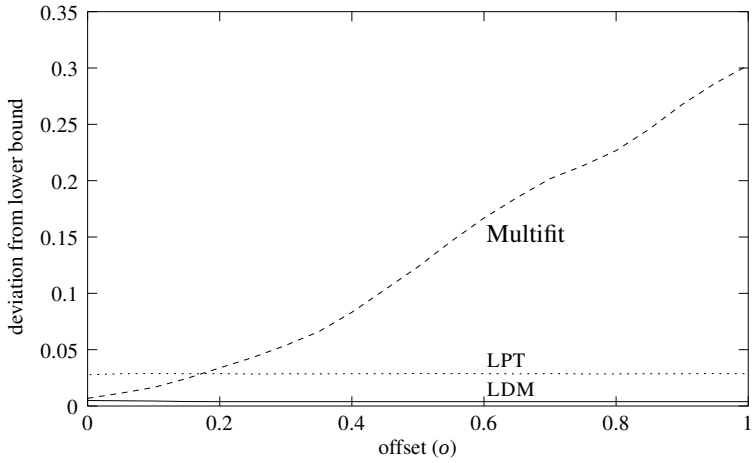

Figure 4. Average deviation of the cost of partitions given by LDM, LPT, and Multifit from the lower bound $\max(a_n, S(A)/m)$, where $m = 10$, $n = 100$, and the item sizes are uniformly distributed over $[o, o+1]$. We generated 10,000 problem instances for each choice of offset o.

auxiliary results. The first lemma states that if the item sizes are not too small when expressed as a fraction of the optimal cost, then LDM is optimal.

Lemma 1. *Let I be an instance of multiprocessor scheduling with item sizes $a_i > f_I^*/3$. Then* LDM *returns an optimal partition.*

Proof. If $n \leq m$, then LDM returns the optimal partition in which each item is assigned to a different subset. Suppose, on the other hand, that $n > m$. As each item size $a_i > f_I^*/3$, at most two items are assigned to the same subset in an optimal partition. Hence, I contains at most $2m$ items, i.e., $m < n \leq 2m$. Furthermore, it can be verified that an optimal partition $\mathcal{A}^*$ is obtained by assigning the largest m items increasingly to the subsets and the remaining $n-m$ items decreasingly. Thereby, the $2m-n$ subsets with the largest items do not get a second item assigned. Formally, $A_i^* = \{n-m+i, n-m-i+1\}$ for $1 \leq i \leq n-m$ and $A_i^* = \{n-m+i\}$ for $n-m < i < m$. In the remainder of the proof, we show that this partition $\mathcal{A}^*$ is obtained by LDM.

It can be verified that the first $m-1$ differencing operations of LDM construct the partial solution $\mathcal{A}_{n+m-1} = a_{n-m+1}$ - a_{n-m+2} - ... - a_n from the initial solutions $\mathcal{A}_{n-m+1}, \mathcal{A}_{n-m+2}, \ldots, \mathcal{A}_n$. Now, the so-called first phase of LDM starts. Let t be the number of iterations it takes LDM *before* selecting the solution $\mathcal{A}_{n+m-1}$ after it is constructed. As I contains at most $2m$ items, we have $t < m$. This implies that $\mathcal{A}' = a_{n-m-t}$ - $a_{n-m-t+1}$ - ... - a_{n-m} is derived during these iterations. We claim that $\mathcal{A}_{n+m-1}$ is next differenced with $\mathcal{A}'$. For $t = m-1$ this is true as $\mathcal{A}_{n+m-1}$ and $\mathcal{A}'$ are the only solutions in L. For $0 \leq t < m-1$, the statement holds as $d(\mathcal{A}') = a_{n-m} > d(\mathcal{A}_{n+m-1})$. This ends the first phase of LDM. More general, we define a phase as $t+1$ successive iterations of LDM for some $t \geq 0$ in which a partition $\mathcal{A}' = a_{i-t}$ - a_{i-t+1} - ... - a_i is

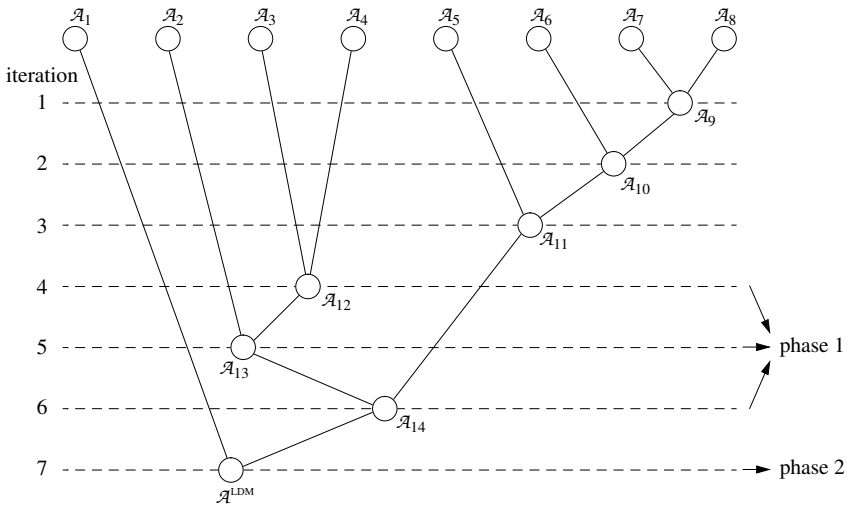

Figure 5. The partition of the last four iterations of LDM into two phases for a possible run of LDM in the case that $m = 4$ and $n = 8$.

constructed and in which $\mathcal{A}'$ is differenced with a partition originating from $\mathcal{A}_{n+m-1}$; see Figure 5. It can be verified that LDM proceeds in phases until $\mathcal{A}^{\text{LDM}}$ is obtained.

We prove by induction on i that the partial solution $\mathcal{A}$ derived in phase i *conforms* to $\mathcal{A}^*$, which by definition means that $A_j \subseteq A_j^*$ for all $1 \leq j \leq m$.

For $i = 1$, the induction hypothesis trivially holds. Next, assume $i \geq 2$ and let $\mathcal{A}$ be obtained by differencing $\mathcal{A}_{\text{phase}}$ determined in the previous phase, i.e., phase $i-1$, and a_{x-t} - a_{x-t+1} - ... - a_x with $n - 2m \leq x \leq n - m$ and $t \geq 0$. As $\mathcal{A}_{\text{phase}}$ conforms to $\mathcal{A}^*$ by the induction hypothesis, we have that $\mathcal{A}$ conforms to $\mathcal{A}^*$ if a_{x-t} is assigned to the smallest subset in $\mathcal{A}_{\text{phase}}$ containing only one item, a_{x-t+1} to the second smallest subset containing only one item, and so on. This means that the induction hypothesis is only violated if assigning some item j to a subset in $\mathcal{A}_{\text{phase}}$ that already contains two items larger than j results in a subset V satisfying $S(V) \leq S(A^*(j))$, where $A^*(j)$ is the subset in $\mathcal{A}^*$ containing item j. However, this would imply that $a_j \leq f_I^*/3$, which yields a contradiction. Hence, the induction hypothesis holds and therefore the lemma. □

The following result can roughly be interpreted as that in each iteration LDM constructs a solution that is better than the worst of the two solutions from which it is constructed. The result is already proved by Karmarkar and Karp [13]. For completeness we include its proof.

Lemma 2. *Let partition $\mathcal{A}$ be obtained by differencing partitions $\mathcal{A}'$ and $\mathcal{A}''$. We then have $d(\mathcal{A}) \leq \max(d(\mathcal{A}'), d(\mathcal{A}''))$.*

Proof. Let A_i and A_j be the subsets in $\mathcal{A}$ with maximum and minimum sum, respectively.

Without loss of generality, we assume $A_i = A'_i \cup A''_i$ and $A_j = A'_j \cup A''_j$. Now,

$$\begin{aligned} d(\mathcal{A}) &= S(A'_i) + S(A''_i) - S(A'_j) - S(A''_j) \\ &= (S(A'_i) - S(A'_j)) + (S(A''_i) - S(A''_j)). \end{aligned}$$

One of the two terms in the last expression is non-negative and the other is non-positive. Furthermore, the first term is bounded by $d(\mathcal{A}')$, whereas the second one is bounded by $d(\mathcal{A}'')$. This proves the lemma. □

We use this result to prove the following lemma.

Lemma 3. *Let α be the largest item with $a_\alpha \leq f_I^*/3$ for a given multiprocessor scheduling instance I. Partition $\mathcal{A}^{\text{LDM}}$ is optimal if α does not exist. Otherwise, $\mathcal{A}^{\text{LDM}}$ is either optimal or $d(\mathcal{A}^{\text{LDM}}) \leq a_\alpha$.*

Proof. The case that α does not exist is implied by Lemma 1. Next, assume that α exists. As a problem instance can have at most $2m$ items with size larger than $f_I^*/3$, we have $\alpha \geq n - 2m$. If at any moment during the execution of LDM list L only contains partial solutions $\mathcal{A}$ with $d(\mathcal{A}) \leq a_\alpha$, then Lemma 2 yields $d(\mathcal{A}^{\text{LDM}}) \leq a_\alpha$. In the remainder of the proof, we show that LDM returns an optimal partition if this is not the case.

Let $\mathcal{A}(i)$ be the last solution in L at the start of iteration i, where we assume that the partial solutions in list L are in non-decreasing order of the value of d. Note that by assumption $d(\mathcal{A}(i)) > a_\alpha$ for all $i \geq 1$. Consider iteration i_0 in which solution $\mathcal{A}_\alpha$ is selected. Then the last two solutions in list L are $\mathcal{A}(i_0)$ and $\mathcal{A}_\alpha$ with $d(\mathcal{A}_\alpha) = a_\alpha$. During the first $i_0 - 1$ iterations, LDM only operates on the initial solutions $\mathcal{A}_{\alpha+1}, \mathcal{A}_{\alpha+2}, \ldots, \mathcal{A}_n$ and on solutions obtained from them. This implies that L still contains the initial solutions $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_\alpha$. As it takes at least m items larger than α to construct a solution $\mathcal{A}$ with $d(\mathcal{A}) \leq d(\mathcal{A}_\alpha)$ and as $\alpha \geq n - 2m$, list L contains at most one additional solution $\mathcal{A}$. This means that we either have $L = \mathcal{A}_1; \mathcal{A}_2; \ldots; \mathcal{A}_{\alpha-1}; \mathcal{A}_\alpha; \mathcal{A}(i_0)$ or $L = \mathcal{A}_1; \mathcal{A}_2; \ldots; \mathcal{A}_{l-1}; \mathcal{A}; \mathcal{A}_l; \ldots; \mathcal{A}_{\alpha-1}; \mathcal{A}_\alpha; \mathcal{A}(i_0)$ for some $1 \leq l < \alpha$, where $\mathcal{A}$ only contains items larger than α.

Now, LDM proceeds by each time differencing the solution obtained in the last iteration with the last solution in the list as solution $\mathcal{A}(i)$ always satisfies $d(\mathcal{A}(i)) > a_\alpha$ by assumption. Remark that LDM finishes after $i_0 + \alpha$ or $i_0 + \alpha + 1$ iterations depending on whether $\mathcal{A}$ exists.

Let i_{end} be the last iteration of LDM and let $\mathcal{A}(i_{\text{end}} + 1)$ be defined as $\mathcal{A}^{\text{LDM}}$. Furthermore, we define $W(i)$ for $i_0 \leq i \leq i_{\text{end}} + 1$ such that $j \in W(i)$ if and only if subset $A_j(i)$ from $\mathcal{A}(i)$ is more than a_α larger than the minimum subset sum, i.e., $S(A_j(i)) > \min_{j'} S(A_{j'}(i)) + a_\alpha$. As by assumption $d(\mathcal{A}(i)) > a_\alpha$, set $W(i)$ is not empty.

We prove by induction on i that for all $j \in W(i)$, we have that $A_j(i)$ only contains items at least $\alpha + 1$ and that $S(A_j(i)) \leq f_I^*$ for all $j \in W(i)$. As $W(i)$ contains the subsets with largest sum and because $W(i)$ is not empty, this proves the optimality of $\mathcal{A}^{\text{LDM}}$.

Consider $i = i_0$. LDM does not operate on the initial solutions $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_\alpha$ during the first i_0 iterations. Hence, the maximum subset sum in $\mathcal{A}(i_0)$ is at most the maximum subset sum in the partition given by LDM for the instance I', which by definition is obtained from I by removing all items with size at most $f_I^*/3$. As $f_{I'}^* \leq f_I^*$ and by Lemma 1, we now get that the maximum subset sum in $\mathcal{A}(i_0)$ is at most f_I^*, which proves the induction hypothesis for $i = i_0$.

Next, assume that $i_0 < i \leq i_{\text{end}} + 1$. First, consider the case that $\mathcal{A}(i)$ is obtained by differencing $\mathcal{A}(i-1)$ with an initial solution A_p. Then item p is added to the subset $A_j(i-1)$ in $\mathcal{A}(i-1)$ with minimum sum. As a result of this operation, neither subset j nor any other subset is added to W, since $a_p \leq a_\alpha$. Hence, no items are added to the subsets in $W(i)$ and $W(i) \subseteq W(i-1)$. The induction hypothesis now follows from the case $i-1$.

Suppose, on the other hand, that $\mathcal{A}(i-1)$ is differenced with $\mathcal{A}$. The larger the sum of a subset in $\mathcal{A}(i-1)$, the smaller the sum of the subset from $\mathcal{A}$ that is added to it. Furthermore, the difference in subset sum in $\mathcal{A}$ is not larger than a_α. Hence, $W(i) \subseteq W(i-1)$. By definition, the subsets in $W(i-1)$ are the subsets from $\mathcal{A}(i-1)$ with largest sum. Furthermore, as by the induction hypothesis only items at least $\alpha+1$ are assigned to these subsets, we have that they are also contained in $\mathcal{A}(i_0)$. Hence, each subset $A_j(i)$ from $\mathcal{A}(i)$ with $j \in W(i) \subseteq W(i-1)$ is also present in the solution obtained by differencing $\mathcal{A}(i_0)$ with $\mathcal{A}$. Now, Lemma 1 yields $S(A_j(i)) \leq f^*_{I'}$, where again I' is as defined above. As $f^*_{I'} \leq f^*_I$, we get $S(A_j(i)) \leq f^*_I$, which proves the induction hypothesis. □

Using Lemmas 1 and 3, we can prove bounds on the performance ratio of LDM as a function of m. Note that the upper bound given for m equals the lower bound for $m+1$.

Theorem 1. *The performance ratio R of* LDM *is 7/6 for $m = 2$. For $m \geq 3$, it satisfies*

$$\frac{4}{3} - \frac{1}{3(m-1)} \leq R \leq \frac{4}{3} - \frac{1}{3m}.$$

Proof. First, we show that $4/3 - 1/(3m)$ is a performance bound for any $m \geq 2$. Note that for $m = 2$ this implies a performance bound of $7/6$. Consider an arbitrary problem instance I of multiprocessor scheduling and assume that $\mathcal{A}^{\text{LDM}}$ is not optimal. We have

$$f^*_I \geq \frac{1}{m} \sum_{j=1}^{m} S(A^{\text{LDM}}_j) \geq \min_{1 \leq j \leq m} S(A^{\text{LDM}}_j) + \frac{d(\mathcal{A}^{\text{LDM}})}{m}$$

and thus

$$\begin{aligned} f_I(\mathcal{A}^{\text{LDM}}) &= \min\nolimits_{1 \leq j \leq m} S(A^{\text{LDM}}_j) + d(\mathcal{A}^{\text{LDM}}) \\ &\leq f^*_I - \tfrac{d(\mathcal{A}^{\text{LDM}})}{m} + d(\mathcal{A}^{\text{LDM}}). \end{aligned}$$

Furthermore, as by assumption $\mathcal{A}^{\text{LDM}}$ is not optimal, Lemma 3 gives $d(\mathcal{A}^{\text{LDM}}) \leq a_\alpha$, where $a_\alpha \leq f^*_I/3$. Combining these results yields the performance bound.

The lower bound $7/6$ on the performance ratio for $m = 2$ holds as LDM may return the partition $3,2,2$ - $3,2$ with objective value 7, while the optimal partition $3,3$ - $2,2,2$ only has cost 6. This problem instance is also used by Fischetti and Martello [4].

For the lower bound $4/3 - 1/(3(m-1))$ when $m \geq 3$ consider the problem instance for which the optimal partition with objective value $3(m-1)$ is given by $A^*_1 = 3(m-1)$, $A^*_2 = (m-1, m-1, m-1)$, $A^*_{2j-1} = (2(m-1) - j + 1, m + j - 2)$ with $2 \leq j \leq (m+1)/2$ and $A^*_{2j} = (2(m-1) - j + 1, m + j - 2)$ with $2 \leq j \leq m/2$; see Figure 6. Note that the sum of each subset is $3(m-1)$. We now derive $\mathcal{A}^{\text{LDM}}$, where we assume that m is odd. The case that m is even can be handled similarly. In the first $m-1$ iterations LDM

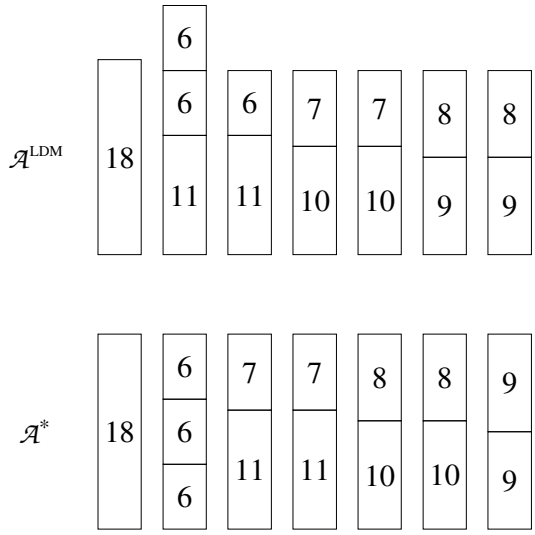

Figure 6. Instance with a performance ratio of $4/3 - 1/(3(m-1)) = 23/18$ for $m = 7$.

constructs the partial solution $\mathcal{A}$ with the largest m items all assigned to a different subset, i.e., $\mathcal{A} = 3(m-1)$ - $2(m-1)-1$ - $2(m-1)-1$ - $2(m-1)-2$ - $2(m-1)-2$ - ... - $3/2(m-1)$. Note that $d(\mathcal{A}) = 3/2(m-1)$. In the next iteration, LDM differences $\mathcal{A}$ and $\mathcal{A}_m$, where $\mathcal{A}_m$ with $d(\mathcal{A}_m) = 3/2(m-1)-1$ contains the next largest item, i.e., item m with size $3/2(m-1)-1$. The differencing operation adds item m to the subset containing $3/2(m-1)$ resulting in a subset with sum $3(m-1)-1$.

This process is repeated in the last $m-1$ iterations. More precisely, since at the start of each of the following iterations $i = m+1, m+2, \ldots, 2m-2$ we have $d(\mathcal{A}) = d(\mathcal{A}_i) + 1$, where $\mathcal{A}$ is the solution obtained in iteration $i-1$ and $\mathcal{A}_i$ is the initial solution only containing item i, LDM differences $\mathcal{A}$ with $\mathcal{A}_i$ in iteration i. This is also the case in the last iteration $i = 2m+1$ as then only two solutions remain. As a result, $A_1^{\text{LDM}} = 3(m-1)$, $A_2^{\text{LDM}} = (2(m-1)-1, m-1, m-1)$, $A_3^{\text{LDM}} = (2(m-1)-1, m-1)$ and A_{2j}^{LDM} and A_{2j+1}^{LDM} are both given by $(2(m-1)-j, (m-1)+j-1)$ with $2 \leq j \leq (m-1)/2$. Hence, the sum of the first two subsets is $3(m-1)$ and $4(m-1)-1$, respectively, whereas the sum of each other subset is $3(m-1)-1$. Consequently, $f_I(\mathcal{A}^{\text{LDM}}) = 4(m-1)-1$ implying a performance ratio of $4/3 - 1/(3(m-1))$. □

4 Performance ratio as a function of both m and n

In this section, we consider the question whether we can improve the performance guarantee of LDM if we have additional information about the number of items. For $m = 2$ the question is answered in Theorem 2, whereas Theorem 3 settles the case $m \geq 3$. Figure 7 shows the result of Theorem 3 for $m = 10$. The results show that if we cannot exclude the possibility that the subsets in a partition contain more than two items on average, the performance guarantee cannot be improved.

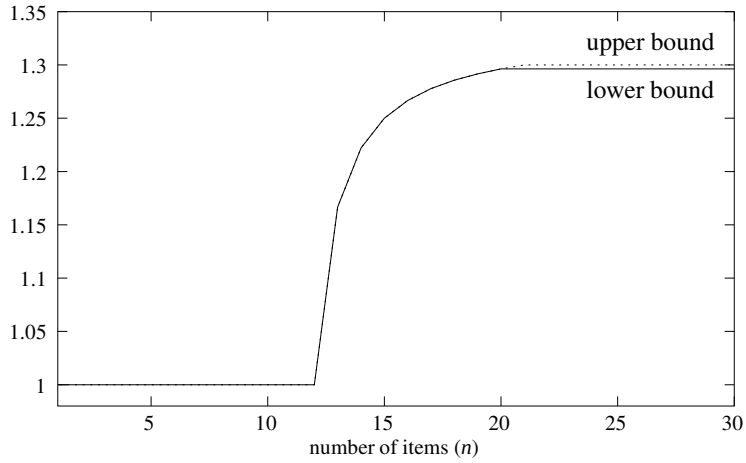

Figure 7. Performance results stated by Theorem 3 for $m = 10$.

Theorem 2. *Let $m = 2$. Then* LDM *is optimal for $n \leq 4$ and* LDM *has performance ratio of $7/6$ for any given $n \geq 5$.*

Proof. Fischetti and Martello [4] prove that LDM is optimal for $n \leq 4$. Next, let $n \geq 5$ be given. In the proof of Theorem 1 we give an instance consisting of five items for which the performance ratio is $7/6$. Furthermore, adding $n-5$ items with size zero to this instance does not affect its performance ratio. Hence, $7/6$ is a lower bound on the performance ratio. As Theorem 1 states that $7/6$ is also a performance bound, we conclude that $7/6$ is a performance ratio for any given $n \geq 5$. □

Theorem 3. *For given $m \geq 3$, the performance ratio R of* LDM *satisfies*

$$\begin{aligned} R &= 1 && \text{if } n \leq m+2 \\ R &= \tfrac{4}{3} - \tfrac{1}{3(n-m-1)} && \text{if } m+2 \leq n \leq 2m \\ \tfrac{4}{3} - \tfrac{1}{3(m-1)} \leq R &\leq \tfrac{4}{3} - \tfrac{1}{3m} && \text{if } n > 2m. \end{aligned}$$

Proof. LDM is optimal for $n \leq m$ as it returns the optimal partition in which each item is assigned to a different subset. The algorithm is also optimal for $n = m+1$ as in that case only the two smallest items are in the same subset. The optimality for $n = m+2$ is implied by the claimed performance ratio for $m+2 \leq n \leq 2m$ as it equals 1 for this case. Before proving the case $m+2 \leq n \leq 2m$, we first consider $n > 2m$. Then, the upper bound on R follows directly from Theorem 1. Consider the claimed lower bound on R. As the problem instance presented in the proof of Theorem 1 with performance ratio $4/3 - 1/(3(m-1))$ consists of $2m$ items, the lower bound holds for $n = 2m$. Furthermore, since adding items with zero size does not change the performance ratio of the instance, we get that the lower bound holds for any $n \geq 2m$.

In the remainder of the proof we focus on the case that $m+2 \leq n \leq 2m$. Consider again the problem instance I given in the proof of Theorem 1 for the case that we have to partition the items into $m' = n-m$ subsets. Note that $m' \leq m$. Then I consists

of $2m'$ items and it has a performance ratio of $4/3 - 1/(3(m'-1))$. Adding $n-2m'$ items with size $3(m'-1)$ to the instance results in an instance with n items and with the same performance ratio of $4/3 - 1/(3(m'-1))$. As $m' = n-m$, this proves that $4/3 - 1/(3(n-m-1))$ is a lower bound on R.

We now prove that $4/3 - 1/(3(n-m-1))$ is also an upper bound on R by showing that it is a performance bound for an arbitrary problem instance I for which LDM does not return an optimal partition. Note that by Lemma 1, we can assume that $a_1 \leq f_I^*/3$. We first derive an upper bound on $f_I(\mathcal{A}^{\text{LDM}})$ and next we derive a lower bound on f_I^*. Combining these results yields the claimed performance bound.

During the first m iterations, LDM constructs solution $\mathcal{A} = a_{m'+1}$ - $a_{m'+2}$ - $\ldots a_n$ in which the largest m items are each assigned to a different subset. As $n \leq 2m$, LDM next proceeds in phases as described in the proof of Lemma 1. By induction on the number of executed phases, we can easily prove the following properties.

- Only the subsets $A_1, A_2, \ldots, A_{m'}$ in $\mathcal{A}$ containing one of the smallest m' items can receive additional items.
- The only situation in which $A_{m'}$ receives more than one item is when the m' smallest items are decreasingly assigned to the subsets $A_1, A_2, \ldots, A_{m'}$, i.e., item i with $1 \leq i \leq m'$ is assigned to $A_{m'-i+1}$. As a result, each of the subsets $A_1, A_2, \ldots, A_{m'}$ has a cardinality of two.

Let the subsets in $\mathcal{A}^{\text{LDM}}$ be numbered, such that $A_j \subseteq A_j^{\text{LDM}}$ for all $1 \leq j \leq m$. As by assumption LDM does not return an optimal partition, the sum of a subset in $\mathcal{A}^{\text{LDM}}$ containing only one item is not maximal. Hence, if no second item is assigned to $A_{m'}$, then

$$f_I(\mathcal{A}^{\text{LDM}}) \leq \min_{1 \leq j < m'} S(A_j^{\text{LDM}}) + \max(d_1^{m'-1}(\mathcal{A}^{\text{LDM}})), \tag{1}$$

where $d_1^{m'-1}(\mathcal{A}^{\text{LDM}})$ is defined as the difference between the minimum and maximum subset sum of the subsets $A_1^{\text{LDM}}, A_2^{\text{LDM}}, \ldots, A_{m'-1}^{\text{LDM}}$. Next, assume that a second item is assigned to $A_{m'}$, i.e., $A_{m'}^{\text{LDM}}$ contains the items $2m'$ and 1. Then

$$f_I(\mathcal{A}^{\text{LDM}}) \leq \min_{1 \leq j < m'} S(A_j^{\text{LDM}}) + \max(d_1^{m'-1}(\mathcal{A}^{\text{LDM}}), a_1) \tag{2}$$

holds if $a_{2m'}$ is at most the sum of any other subset in $\mathcal{A}^{\text{LDM}}$ to which two items are assigned. Formally,

$$a_x \leq \min_{1 \leq j < m'} S(A_j^{\text{LDM}}). \tag{3}$$

Before using Equations 1 and 2 to prove that $4/3 - 1/(3(n-m-1))$ is a performance bound, we prove (3).

Clearly, item 1 is added to the subset containing item $2m'$ by LDM during the last phase of its execution, where solution $\mathcal{A}'$ containing $A_{m'} = a_{2m'}$ is differenced with $\mathcal{A}'' = a_1$ - a_2 - $\ldots$ - a_l for some $l \leq m$. If $l = 1$ then $\mathcal{A}^{\text{LDM}}$ is obtained from $\mathcal{A}'$ by assigning a_1 to the subset with minimum sum. As by assumption this is the subset containing $a_{m'}$, we get (3). Suppose, on the other hand, that $l > 1$ and that (3) does not hold. We show that this yields a contradiction. As for $\mathcal{A}^{\text{LDM}}$, let the subsets in $\mathcal{A}'$ be numbered such that $A_j \subseteq A'_j$. As (3) does not hold, we have $\min_{1 \leq j \leq m'} S(A'_j) + a_l < a_{2m'}$. Furthermore, since $S(A'_{m'}) = a_{2m'}$, we have that $d(\mathcal{A}') \geq a_{2m'} - \min_{1 \leq j < m'} S(A'_j)$. Combining these

observations yields that $a_l < d(\mathcal{A}')$. However, this implies that LDM does not add item l to $\mathcal{A}'$ via $\mathcal{A}''$, but directly via the initial solution $\mathcal{A}_l$, which gives a constradiction. This proves (3) and consequently also (2).

We now derive a lower bound on f_I^*. Consider an optimal partition $\mathcal{A}^*$. Without loss of generality, we can assume that $A^*_{m'+1}, A^*_{m'+2}, \ldots, A^*_m$ each contain exactly one item and that these are the largest $n - 2m'$ items. The other $2m'$ items are assigned to the subsets $A^*_1, A^*_2, \ldots, A^*_{m'}$. If each of these latter subsets contains two items, than $\mathcal{A}^*$ is obtained by assigning the largest m' items increasingly to the subsets and the remaining m' items decreasingly. Along the same lines as in the proof of Lemma 1, it can be shown that in this case LDM returns an optimal partition.

Assume that at least one of the subsets $A^*_1, A^*_2, \ldots, A^*_{m'}$ contains only one item. Without loss of generality, we assume that this is $A^*_{m'}$ and that it contains the largest item not already assigned to $A^*_{m'+1}, A^*_{m'+2}, \ldots, A^*_m$. Now, the subsets $A^*_1, A^*_2, \ldots, A^*_{m'-1}$ contain the items from $A^{\mathrm{LDM}}_1, A^{\mathrm{LDM}}_2, \ldots, A^{\mathrm{LDM}}_{m'}$ minus item $2m'$. As a result,

$$f_I^* \geq \min_{1 \leq j < m'} S(A_j^{\mathrm{LDM}}) + \frac{d_1^{m'-1}(\mathcal{A}^{\mathrm{LDM}}) + a_1}{m' - 1}$$

if $A^{\mathrm{LDM}}_{m'}$ contains item 1 and

$$f_I^* \geq \min_{1 \leq j < m'} S(A_j^{\mathrm{LDM}}) + \frac{d_1^{m'-1}(\mathcal{A}^{\mathrm{LDM}})}{m' - 1},$$

otherwise. As mentioned above, $a_1 \leq f_I^*/3$. Furthermore, we have by Lemma 3 that $d_1^{m'-1}(\mathcal{A}^{\mathrm{LDM}}) \leq f_I^*/3$. Combining these results with (2) and (1) gives that $f_I(\mathcal{A}^{\mathrm{LDM}}) \leq (4/3 - 1/(3m' - 1)) f_I^*$, which proves the theorem. □

References

1. E.G. Coffman Jr., G.N. Frederickson, and G.S. Lueker. A note on expected makespans for largest-first sequences of independent tasks on two processors. *Mathematics of Operations Research*, 9:260–266, 1984.
2. E.G. Coffman Jr, M.R. Garey, and D.S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7(1):1–17, 1978.
3. E.G. Coffman Jr. and W. Whitt. Recent asymptotic results in the probabilistic analysis of schedule makespans. In P. Chretienne, E.G. Coffman Jr., J.K. Lenstra, and Z. Liu, editors, *Scheduling Theory and its Applications*, pages 15–31. Wiley, 1995.
4. M. Fischetti and S. Martello. Worst-case analysis of the differencing method for the partition problem. *Mathematical Programming*, 37:117–120, 1987.
5. J.B.G. Frenk and A.H.G. Rinnooy Kan. The rate of convergence to optimality of the LPT rule. *Discrete Applied Mathematics*, 14:187–197, 1986.
6. D.K. Friesen. Tighter bounds for the multifit processor scheduling algorithm. *SIAM Journal on Computing*, 13(1):170–181, 1984.
7. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
8. I.P. Gent and T. Walsh. Phase transitions and annealed theories: Number partitioning as a case study. In *Proceedings of the* 12^{th} *European Conference on Artifi cial Intelligence*, pages 170–174, Denver, Colorado, 1996.

9. R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.
10. 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*, 5:287–326, 1979.
11. B. Hayes. The easiest hard problem. *American Scientist*, 90:113–117, 2002.
12. D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.
13. N. Karmarkar and R.M. Karp. The differencing method of set partitioning. Technical Report UCB/CSD 82/113, University of California, Berkeley, 1982.
14. R.E. Korf. A complete anytime algorithm for number partitioning. *Artifi cial Intelligence*, 106:181–203, 1998.
15. G.S. Lueker. A note on the average-case behavior of a simple differencing method for partitioning. *Operations Research Letters*, 6(6):285–287, 1987.
16. S. Mertens. A complete anytime algorithm for balanced number partitioning, 1999. preprint xxx.lanl.gov/abs/cs.DS/9903011.
17. S. Mertens. A physicist's approach to number partitioning. *Theoretical Computer Science*, 265:79–108, 2001.
18. W. Michiels, J. Korst, E. Aarts, and J. van Leeuwen. Performance ratios for the differencing method applied to the balanced number partitioning problem. In *Proceedings of the* 20^{th} *International Symposium on Theoretical Aspects of Computer Science*, Berlin, 2003.
19. W. Ruml, J.T. Ngo, J. Marks, and S. Shieber. Easily searched encodings for number partitioning. *Journal of Optimization Theory and Applications*, 89(2):251–291, 1996.
20. R.H. Storer, S.W. Flanders, and S.D. Wu. Problem space local search for number partitioning. *Annals of Operations Research*, 63:465–487, 1996.
21. L.H. Tasi. The modified differencing method for the set partitioning problem with cardinality constraints. *Discrete Applied Mathematics*, 63:175–180, 1995.
22. L.H. Tsai. Asymptotic analysis of an algorithm for balanced parallel processor scheduling. *SIAM Journal on Computing*, 21(1):59–64, 1992.
23. B. Yakir. The differencing algorithm LDM for partitioning: A proof of Karp's conjecture. *Mathematics of Operations Research*, 21(1):85–99, 1996.
24. M. Yue. On the exact upper bound for the multifit processor scheduling algorithm. *Annals of Operations Research*, 24:233–259, 1990.