Mathematical Methods of Operations Research (1998) 47:59–82

The k-Partitioning Problem

LUITPOLD BABEL

Institut für Mathematik, Technische Universität München, Arcisstraße 21, D-80290 München, Germany
e-mail: babel@statistik.tu-muenchen.de

HANS KELLERER

Institut für Statistik, Ökonometrie und Operations Research, Universität Graz, Universitätsstraße 15, A-8010 Graz, Austria
e-mail: kellerer@bkfug.kfunigraz.ac.at

VLADIMIR KOTOV

University of Minsk, Faculty of Applied Mathematics and Computer Science, Minsk, 220080, Byelarussia
e-mail: kotov@cadcam.bsu.minsk.by

Abstract: The k-partitioning problem is defined as follows: Given a set of items $\{I_1, I_2, \dots, I_n\}$ where item I_j is of weight $w_j \geq 0$, find a partition $S_1, S_2, \dots, S_m$ of this set with $|S_i| = k$ such that the maximum weight of all subsets S_i is minimal. k-partitioning is strongly related to the classical multiprocessor scheduling problem of minimizing the makespan on identical machines. This paper provides suitable tools for the construction of algorithms which solve exactly the problem. Several approximation algorithms are presented for this NP-hard problem. The worst-case behavior of the algorithms is analyzed. The best of these algorithms achieves a performance bound of 4/3.

Key Words: partition, multiprocessor scheduling, worst-case

1 Introduction

In set partitioning a set of n positive numbers has to be partitioned into m subsets such that the weights, i.e. the sum of the elements of the subsets are equal or at least nearly equal. We consider the following version of this large class of problems:

Let us have a set $\{I_1, I_2, \dots, I_n\}$ of $n = km$ items where item I_j has nonnegative weight $W_j \geq 0$. W.l.o.g. assume the items to be sorted in non-increasing order, i.e. $w_1 \geq w_2 \geq \dots \geq w_n$. The objective is to find a partition $S_1, S_2, \dots, S_m$ of this set with $|S_i| = k$ such that the maximum weight of all subsets S_i is minimal.

© Physica-Verlag 1998

The problem above is a generalization of the well known 3-partitioning problem where $3m$ items of positive integer weight have to be partitioned into m sets, each containing 3 elements, and such that all sets are of equal weight. In [3] the NP-completeness of 3-partitioning has been proved. From this we can deduce that the above defined problem is also hard to solve. Since each subset S_i contains exactly k elements, we call it the k-partitioning problem.

k-partitioning requires that all subsets S_i are of the same cardinality. If this demand is omitted the well-known, classical multiprocessor scheduling problem arises which can be interpreted as a relaxation of the k-partitioning problem.

We have to schedule n independent jobs on m identical, parallel machines. Each job has a certain processing time and can be processed on any of the m machines. Preemptions are not allowed. The objective is to minimize the maximum completion time, i.e. the time when the last job is completed. In scheduling terminology this is the problem $P|\cdot|C_{max}$. Unfortunately, this problem is NP-hard, too (see [3]). For more details and a review on scheduling theory see for example [8].

The constraint that a machine can only process k items can be interpreted as follows: Each machine is equipped with k units of a specific resource. To process one job on one machine, one unit of the resource is necessary.

In the following we identify items I_j with jobs and subsets S_i with machines. w_j is the processing time of job I_j. Then any solution of the scheduling problem corresponds to a partition $S_1, \dots, S_m$ of $\{I_1, \dots, I_n\}$. For $S \subseteq \{I_1, \dots, I_n\}$ let $w(S) = \sum_{I_j \in S} w_j$ be the weight of the set S. For convenience we often identify items with their weights.

We start our considerations in Section 2 with some relations of the k-partitioning problem to the classical multiprocessor scheduling problem $P|\cdot|C_{max}$. Using lower bounds on the value of an optimal solution of the k-partitioning problem, we describe a way to reduce the problem size by splitting the original problem into two subproblems. In Section 3 we propose some heuristic methods and give performance bounds for them. Our best heuristic has a worst-case bound of 4/3.

2 About the Optimum Value of the k-Partitioning Problem

Let C^* and $C^*_{||}$ denote the values of optimal solutions of the k-partitioning problem and the corresponding scheduling problem $P|\cdot|C_{max}$, respectively. Our first result shows that C^* can only be twice as big as $C^*_{||}$.

Theorem 1: For the ratio of C^* and $C^*_{||}$ the inequality

$$\frac{C^*}{C^*_{||}} \leq 1 + \frac{m-1}{m}\,\frac{n-m}{n-1} < 2 - \frac{1}{m}$$

holds. The bound is best possible.

Proof: Let $\hat{S}_1, \hat{S}_2, \dots, \hat{S}_m$ be an optimal solution of $P|\cdot|C_{max}$ with $\hat{S}_i \neq \emptyset$, $i = 1, 2, \dots, m$. We remove from all sets $\hat{S}_i$ with $|\hat{S}_i| > k$ the $|\hat{S}_i| - k$ items of smallest weight and distribute them among the remaining sets in such a way that a partition $S_1, S_2, \dots, S_m$ with $|S_i| = k, i = 1, \dots, m$, results.

W.l.o.g. assume that $w(S_1) = max_{i=1,\dots,m} w(S_i)$. S_1 was created from $\hat{S}_1$ by adding at most $k-1$ items from at most $k-1$ other sets. Let $r_i \geq 0$ be the number of items which were moved from $\hat{S}_i$ to $\hat{S}_1$. The sum of the weights of these items is at most

$$\frac{r_i}{k + r_i}\, w(\hat{S}_i)\ .$$

Let Δ denote the sum of the weights of all items which were added to $\hat{S}_1$. Then

$$\begin{aligned}\Delta &\leq \sum_{i=2}^{m} \frac{r_i}{k+r_i} w(\hat{S}_i) \leq \sum_{i=2}^{m} \frac{r_i}{k+r_i} C^*_{||} \\ &= \left(m - 1 - k \sum_{i=2}^{m} \frac{1}{k+r_i}\right) C^*_{||}\end{aligned}$$

where $r_2 + \dots + r_m \leq k-1$ and $r_2, \dots, r_m \in \mathbb{N}_0$. If the integrality condition is replaced by $r_2, \dots, r_m \geq 0$ then $\sum_{i=2}^{m} \frac{1}{k+r_i}$ is minimal for $r_i = \frac{k-1}{m-1}$. Using $k = \frac{n}{m}$ we get

$$\begin{aligned}\Delta &\leq \left(m - 1 - k \sum_{i=2}^{m} \frac{m-1}{k(m-1)+k-1}\right) C^*_{||} \\ &= (m-1)\frac{k(m-1)+k-1-(m-1)k}{km-1}\, C^*_{||} \\ &= (m-1)\frac{k-1}{mk-1}\, C^*_{||} \\ &= \frac{m-1}{m}\,\frac{n-m}{n-1}\, C^*_{||}\ .\end{aligned}$$

Thus

$$w(S_1) = w(\hat{S}_1) + \Delta \leq \left(1 + \frac{m-1}{m}\frac{n-m}{n-1}\right) C^*_{\|} .$$

With $C^* \leq w(S_1)$ the statement follows.

An example shows that the bound is tight. Let n, m be such that $m-1$ divides $n-1$ (i.e. $(m-1)|(n-1)$) and $w_1 = 1, w_i = \frac{m-1}{n-1}, i = 2, \ldots, n$. An optimal solution of $P| \cdot |C_{max}$ is

$$\hat{S}_1 = \{1\} , \quad \hat{S}_i = \left\{\frac{m-1}{n-1}, \ldots, \frac{m-1}{n-1}\right\} , \quad i = 2, \ldots, m ,$$

with $w(\hat{S}_i) = 1$, $i = 1, \ldots, m$, thus $C^*_{\|} = 1$. An optimal solution of the k-partitioning problem is

$$S_1 = \left\{1, \frac{m-1}{n-1}, \ldots, \frac{m-1}{n-1}\right\}, \quad S_i = \left\{\frac{m-1}{n-1}, \ldots, \frac{m-1}{n-1}\right\}, \quad i = 2, \ldots, m ,$$

with $|S_i| = k = \dfrac{n}{m}$ and

$$w(S_1) = 1 + (k-1)\frac{m-1}{n-1} = 1 + \frac{m-1}{m}\frac{n-m}{n-1} ,$$

$$w(S_i) = k\,\frac{m-1}{n-1} = \frac{m-1}{m}\,\frac{n}{n-1} , \quad i = 2, \ldots, m .$$

Thus

$$C^* = 1 + \frac{m-1}{m}\frac{n-m}{n-1} . \qquad \blacksquare$$

The proof of Theorem 1 provides a method to construct an acceptable solution of the partitioning problem from an optimal solution of the scheduling problem $P| \cdot |C_{max}$. However, since the relaxation is not easier to solve than the original problem, this relation is more of theoretical interest. Another quite obvious connection is helpful especially to solve k-partitioning problems optimally.

Each k-partitioning problem can easily be transformed into a scheduling problem $P| \cdot |C_{max}$. For that purpose define $\hat{w}_i := w_i + M$ with $M > \sum_{i=1}^{n} w_i$ (M

can be interpreted as very large machine setup times, which occur if a machine switches from one job to another). If $\hat{S}_1, \ldots, \hat{S}_m$ is an optimal solution of $P|\cdot|C_{max}$ with processing times $\hat{w}_i$ then obviously $|\hat{S}_i| = k, i = 1, \ldots, m$. Replacing $\hat{w}_i$ by w_i gives an optimal solution $S_1, \ldots, S_m$ of the k-partitioning problem. The value of an optimal solution is $C^* = C^*_{||} - kM$. This shows that the k-partitioning problem can be solved by any algorithm for the scheduling problem $P|\cdot|C_{max}$.

Due to the complexity status of both problems, it is hardly possible to solve large problems exactly. Therefore we are interested in lower bounds, heuristic solutions and techniques to reduce the problem size.

The first lower bound for the k-partitioning problem is a generalization from the two lower bounds

$$C^* \geq w_1 + w_{n-k+2} + \cdots + w_{n-1} + w_n \tag{1}$$

and

$$C^* \geq \frac{1}{m} \sum_{i=1}^{n} w_i \, . \tag{2}$$

and is defined by

$$a := \max_{\ell=1,\ldots,m} \frac{1}{\ell} \left(\sum_{i=1}^{\ell} w_i + \sum_{i=1}^{\ell(k-1)} w_{n-i+1} \right) . \tag{3}$$

For the proof we refer to Lemma 2.

The second lower bound is based on the well-known *Longest Processing Time* (LPT) algorithm for $P|\cdot|C_{max}$. Whenever a machine becomes free for assignment, LPT assigns the largest unexecuted job to that machine. LPT has been first analyzed in [5] and has a worst-case ratio of 4/3 − 1/3m. Let ρ be the smallest index such that at most two items of $I_1, \ldots, I_\rho$ are assigned to a machine by the LPT rule. It follows that $m + 1 \leq \rho \leq 2m$. For each $j \in \{m+1, \ldots, \rho\}$ let the corresponding partition of $\{I_1, \ldots, I_j\}$ produced by LPT be denoted by $S_1^j, \ldots, S_m^j$ ($|S_i^j| \leq 2, i = 1, \ldots, m$). W.l.o.g. we assume that $w(S_1^j) \geq w(S_2^j) \geq \cdots \geq w(S_m^j)$. Set

$$\lambda_\ell := \sum_{i=1}^{\ell} |S_i^j| \quad (\ell = 1, \ldots, m)$$

and for $j = m+1, \dots, p$

$$\hat{b}(j) := \max_{\ell=1,\dots,m} \frac{1}{\ell}\left(\sum_{i=1}^{\ell} w(S_i^j) + \sum_{i=1}^{\ell k-\lambda_\ell} w_{n-i+1}\right) . \tag{4}$$

Furthermore denote for $j = m+1, \dots, p$

$$b(j) := \min\{\hat{b}(j), w_{j-2} + w_{j-1} + w_j\} \tag{5}$$

and

$$b := \max_{j=m+1,\dots,p} b(j) . \tag{6}$$

Finally, define

$$c := w_{p-1} + w_p + w_{p+1} . \tag{7}$$

All these lower bounds are summarized in Lemma 2.

Lemma 2: The optimum solution C^* of the k-partitioning problem satisfies

$$C^* \geq \mathit{level} := \max\{a, b, c\}$$

Proof: In order to show that $C^* \geq a$ let $\ell \in \{1, 2, \dots, m\}$ and define J_ℓ to be the set containing the ℓ items of largest and the $\ell(k-1)$ items of smallest weight of the set $\{I_1, \dots, I_n\}$, i.e. $J_\ell = \{I_1, \dots, I_\ell, I_{n-\ell(k-1)+1}, \dots, I_n\}$. We consider the k-partitioning problem for the set J_ℓ (now J_ℓ is partitioned into ℓ sets of cardinality k). Let $C^*(J_\ell)$ denote the value of an optimal solution.

Let $S_1, \dots, S_m$ be an optimal solution of the original problem. Select ℓ sets, say $S_1, \dots, S_\ell$ with $\{I_1, \dots, I_\ell\} \subseteq S_1 \cup \dots \cup S_\ell$. Replace each item $I_p \in S_1 \cup \dots \cup S_\ell$ with $I_p \notin J_\ell$ by an item $I_q \in S_{\ell+1} \cup \dots \cup S_m$ with $I_q \in J_\ell$. The new partition $\tilde{S}_1, \dots, \tilde{S}_\ell$ fulfills $\tilde{S}_1 \cup \dots \cup \tilde{S}_\ell = J_\ell$.

Since $w_q \leq w_p$ we have $w(\tilde{S}_i) \leq w(S_i)$ and therefore $w(\tilde{S}_i) \leq C^*$ for $i = 1, \dots, \ell$. Using $C^*(J_\ell) \leq w(\tilde{S}_i), i = 1, \dots, \ell$, we conclude

$$C^*(J_\ell) \leq C^* . \tag{8}$$

Finally in analogy to (2) we get

$$C^*(J_\ell) \geq \frac{1}{\ell} \sum_{I_i \in J_\ell} w_i \, .$$

Since this holds for any ℓ, the assertion follows.

To prove the second bound consider for $j \in \{m+1, \dots, p\}$ the partition of the elements $\{I_1, \dots, I_j\}$ in the optimum solution. If there is a set which contains at least three elements of $\{I_1, \dots, I_j\}$, we get $C^* \geq w_{j-2} + w_{j-1} + w_j$. Otherwise, each set contains at most two items of $\{I_1, \dots, I_j\}$. The proof for $C^* \geq b(j)$ is built now in the same way as for $C^* \geq a$, using the fact that LPT is optimal if in the optimal solution not more than two items are assigned to each set. $C^* \geq b$ is a direct consequence.

It is straightforward to see that $C^* \geq c$. For a more general proof we refer to [1]. ■

Let $\ell \in \{1, 2, \dots, m\}$. We split the item set into two sets J_ℓ and $\bar{J}_\ell$ with $J_\ell = \{I_1, \dots, I_\ell, I_{n-\ell(k-1)+1}, \dots, I_n\}$ and $\bar{J}_\ell = \{I_1, \dots, I_n\} - J_\ell$ and consider the k-partitioning problems for J_ℓ and $\bar{J}_\ell$. Any solution of the first problem is a partition into ℓ sets, any solution of the second problem is a partition into $m - \ell$ sets. All sets are of cardinality k.

Let *lowbound*(J_ℓ) and *upbound*$(\bar{J}_\ell)$ denote lower and upper bounds on the values $C^*(J_\ell)$ and $C^*(\bar{J}_\ell)$ of optimal solutions. We may use Lemma 2 to show the following:

Theorem 3: If lowbound$(J_\ell) \geq$ *upbound*$(\bar{J}_\ell)$, *then* $C^* = C^*(J_\ell)$.

Proof: Since *upbound*$(\bar{J}_\ell) \leq$ *lowbound*$(J_\ell) \leq C^*(J_\ell)$ there is a partition of $\bar{J}_\ell$ into $m - \ell$ sets, say $S_{\ell+1}, \dots, S_m$, with $w(S_i) \leq C^*(J_\ell), i = \ell + 1, \dots, m$. Let $S_1, \dots, S_\ell$ be an optimal solution of the k-partitioning problem of J_ℓ. The combination of both partitions provides a partition $S_1, \dots, S_m$ of $\{I_1, \dots, I_n\}$ which satisfies $w(S_i) \leq C^*(J_\ell), i = 1, \dots, m$. Therefore

$$C^* \leq C^*(J_\ell)$$

holds. Using

$$C^* \geq C^*(J_\ell)$$

from (8) we get

$$C^* = C^*(J_\ell) \, .$$

■

A trivial upper bound for $C^*(\bar{J}_\ell)$ is $\sum_{i=1}^{k} w_{\ell+i}$ (the k items of largest weight form a set S_i). More elaborate upper bounds can be obtained by any heuristic methods which are presented in the next section.

Suppose we find a number $\ell < m$ with $lowbound(J_\ell) \geq upbound(\bar{J}_\ell)$. Then it suffices to solve the k-partitioning problem for J_ℓ. An optimal solution for J_ℓ combined with a heuristic solution of $\bar{J}_\ell$ results in an optimal solution of the original problem.

3 Heuristic Methods for the k-Partitioning Problem

3.1 *Folding Algorithm*

Recall that we assume the items to be sorted in nonincreasing order of their weights. We perform a $(k-1)$ time "folding". The resulting sets are

$$S_i = \{I_i, I_{2m-i+1}, I_{2m+i}, I_{4m-i+1}, \ldots, I_{km-i+1}\} \quad \text{if } k \text{ is even}$$

$$S_i = \{I_i, I_{2m-i+1}, I_{2m+i}, I_{4m-i+1}, \ldots, I_{(k-1)m+i}\} \quad \text{if } k \text{ is odd}$$

with $|S_i| = k, i = 1, \ldots, m$.

For $k = 2$, the item of largest weight is combined with the item of smallest weight, the item of second largest weight with the item of second smallest weight, and so on.

Let $C^H = \max_{i=1,\ldots,m} w(S_i)$ denote the value of the solution obtained by the folding algorithm. Our next goal is to establish a worst-case bound in the general case.

Theorem 4: The folding algorithm is optimal for $k = 2$. For $k \geq 3$

$$\frac{C^H}{C^*} \leq 2 - \frac{1}{m}$$

holds. The bound is tight in the general case.

Proof: The case $k = 2$ is obvious. Thus, we assume $k \geq 3$. Let S_p be a set of the folding partition with $w(S_p) \geq w(S_i)$ and S_q a set with $w(S_q) \leq w(S_i)$,

$i = 1, \ldots, m$. If k is even then

$$\begin{aligned} w(S_p) - w(S_q) &= w_p + w_{2m-p+1} + w_{2m+p} + w_{4m-p+1} + \cdots + w_{km-p+1} \\ &\quad - \left(w_q + w_{2m-q+1} + w_{2m+q} + \cdots + w_{km-q+1}\right) \\ &= w_p - \left(w_q - w_{2m-p+1}\right) - \left(w_{2m-q+1} - w_{2m+p}\right) - \cdots \\ &\quad - \left(w_{(k-2)m+q} - w_{km-p+1}\right) - w_{km-q+1} \\ &\leq w_p - w_{km-q+1} \leq w_1 - w_n \,. \end{aligned}$$

Analogously, if k is odd then

$$w(S_p) - w(S_q) \leq w_p - w_{(k-1)m+q} \leq w_1 - w_n \,.$$

For the value C^* of an optimal solution we have

$$C^* \geq w(S_q) + \frac{w(S_p) - w(S_q)}{m}$$

and obviously

$$C^* \geq w_1 \,.$$

Using this inequalities we get

$$\begin{aligned} w(S_p) - C^* &\leq w(S_p) - w(S_q) - \frac{w(S_p) - w(S_q)}{m} \\ &\leq (w_1 - w_n)\left(1 - \frac{1}{m}\right) \\ &\leq C^*\left(1 - \frac{1}{m}\right) \end{aligned}$$

and as a consequence

$$\frac{C^H - C^*}{C^*} = \frac{w(S_p) - C^*}{C^*} \leq \left(1 - \frac{1}{m}\right)\frac{w_1 - w_n}{w_1} \leq 1 - \frac{1}{m} \,.$$

The following example shows that the bound is tight. Let $w_1 = 1, w_i = \frac{1}{2m}$ for $i = 2, \ldots, 2m(m-1)+1$ and $w_i = 0$ for $i = 2m(m-1)+2, \ldots, n$. An optimal solution is given by the partition

$$S_1^* = \{1, 0, \ldots, 0\}\ , \quad w(S_1^*) = 1\ ,$$

$$S_i^* = \left\{\frac{1}{2m}, \ldots, \frac{1}{2m}, 0, \ldots, 0\right\}\ , \quad w(S_i^*) = 1\ , \quad i = 2, \ldots, m\ .$$

The folding algorithm provides

$$S_1 = \left\{1, \frac{1}{2m}, \ldots, \frac{1}{2m}, 0, \ldots, 0\right\}\ , \quad w(S_1) = 2 - \frac{1}{m}\ ,$$

$$S_i = \left\{\frac{1}{2m}, \ldots, \frac{1}{2m}, 0, \ldots, 0\right\}\ , \quad w(S_i) = 1 - \frac{1}{m}\ , \quad i = 2, \ldots, m\ .$$

The worst-case ratio is $2 - \frac{1}{m}$. ■

It follows directly from the proof, that an algorithm which assigns for each $j \in \{1, \ldots, k\}$ the items of the set $\{I_{jm+1}, \ldots, I_{(j+1)m}\}$ arbitrarily by an one to one mapping to the sets $S_1, \ldots, S_m$, has a worst-case bound of $2 - \frac{1}{m}$. Such an algorithm can be implemented in linear time.

Of course, the folding method can also be used to construct a heuristic solution for the scheduling problem $P|\cdot|C_{max}$. Since the proof of the theorem does not use the property that all sets have to be of the same cardinality, we get the same worst-case bound as for the scheduling problem mentioned above.

3.2 List Algorithm

We initialize the sets $S_1, \ldots, S_m$ to be empty. Assume now the items to be listed in *arbitrary* order and assigned one after another to a set S_i of the smallest current weight which contains less than k items. The result is a partition into m sets of cardinality k.

If the cardinality demand is omitted, i.e. if the items are assigned to a set of smallest weight, then we get a heuristic for the scheduling problem $P|\cdot|C_{max}$ which is well known as the *list scheduling* algorithm, the classical on-line algorithm for $P|\cdot|C_{max}$. It has been shown in [4] that its worst-case performance is

$2-\frac{1}{m}$. In analogy to the case of parallel machines we call *modified list scheduling* the corresponding heuristic for k-partitioning.

However, the behavior of modified list scheduling is much worse than list scheduling. The following example indicates that a solution obtained by the list algorithm can be arbitrarily bad. For $k \geq m$ let $w_1 = 1$, $w_i = 0$ for $i = 2, \ldots, n-m+1$ and $w_i = 1$ for $i = n-m+2, \ldots, n$. The list algorithm provides

$$S_1 = \{1, 0, \ldots, 0, 1, \ldots, 1\}\,, \quad w(S_1) = m\,,$$

$$S_i = \{0, \ldots, 0\}\,, \quad w(S_i) = 0\,, \quad i = 2, \ldots, m\,.$$

Since

$$S_i^* = \{1, 0, \ldots, 0\}\,, \quad w(S_i^*) = 1\,, \quad i = 1, \ldots, m$$

is an optimal solution, the worst-case ratio is m.

The reason for the poor performance is the unfavourable order of the items and the fact that a set S_i is blocked as soon as it contains k items. Intuitively, we get an improvement if the items are listed according to nonincreasing weights, i.e. $w_1 \geq w_2 \geq \cdots \geq w_n$. We will call this variant the *modified longest processing time* (MLPT) algorithm, analogous to the corresponding LPT algorithm for $P|\cdot|C_{max}$. It can be easily seen that in the case $k = 2$ MLPT coincides with the folding algorithm, thus it produces an optimal solution. For the general case let C^H denote the value of a solution obtained by the MLPT algorithm. The analysis of MLPT for 3-partitioning is the contents of [7]. It is shown that the worst-case bound remains the same as for LPT.

Theorem 5 ([7]): For $k = 3$ MLPT yields

$$\frac{C^H}{C^*} \leq \frac{4}{3} - \frac{1}{3m}\,.$$

The bound is tight.

The proof of this statement is rather complicated and we do not know how to generalize it to the case $k > 3$. Nevertheless, we assume the worst-case performance should be $4/3 - 1/3m$, as well. Let us mention that recently a heuristic for 3-partitioning with worst-case bound $7/6$ has been found (see [6] which requires only $O(n \log n)$ time).

3.3 Exchange Algorithm

Let $S_1, \dots, S_m$ be a partition of $\{I_1, \dots, I_n\}$ with $|S_i| = k$, $i = 1, \dots, m$, and assume $w(S_p) \geq w(S_i)$, $i = 1, \dots, m$, $w(S_q) < w(S_p)$. A *2-change* is defined in the following way.

Let $I_r \in S_p$, $I_s \in S_q$ with $0 < w_r - w_s < w(S_p) - w(S_q)$. Exchange items I_r and I_s, i.e. let

$$S_p' := S_p - \{I_r\} \cup \{I_s\}$$

$$S_q' := S_q - \{I_s\} \cup \{I_r\}$$

$$S_i' := S_i \,, \quad i \neq p\,, q\,.$$

As a consequence we get either $max_{i=1,\dots,m} w(S_i') < max_{i=1,\dots,m} w(S_i)$ or $max_{i=1,\dots,m} w(S_i') = max_{i=1,\dots,m} w(S_i)$ and the number of sets of largest weight is smaller than before.

The exchange algorithm starts with an arbitrary partition into sets of equal cardinality and improves it by 2-changes as far as possible. A final partition which allows no further 2-changes is called *2-optimal.*

Let C^H denote the value of a 2-optimal partition. Then we get:

Theorem 6: For the exchange algorithm

$$\frac{C^H}{C^*} < 2 - \frac{2}{m+1}$$

holds. The bound is best possible.

Proof: W.l.o.g. we can assume that $C^* = 1$ (if necessary divide all weights w_i by C^*). Therefore for any partition $S_1, \dots, S_m$

$$\sum_{i=1}^{n} w(S_i) = \sum_{i=1}^{n} w_i \leq m \tag{9}$$

holds.

Assume $w(S_1) = 1 + \alpha \geq w(S_i)$ and $w(S_m) \leq w(S_i)$, $i = 1, \dots, m$. Define $\Delta_{1i} = w(S_1) - w(S_i)$ and $\delta_{1i} = min\{w_r - w_s \mid w_r - w_s > 0, I_r \in S_1, I_s \in S_i\}$. If

$S_1, \dots, S_m$ is 2-optimal then α corresponds to the relative error and

$$\Delta_{1i} \le \delta_{1i}\,, \quad i = 1, \dots, m\,. \tag{10}$$

Using (9) we get

$$\sum_{i=2}^{m} \Delta_{1i} = \sum_{i=2}^{m} (w(S_1) - w(S_i)) = mw(S_1) - \sum_{i=1}^{m} w(S_i)$$
$$\ge m(1+\alpha) - m = m\alpha\,.$$

Since $\Delta_{1m} \ge \Delta_{1i}$, $i \ge 2$, we conclude $\Delta_{1m} \ge \frac{m}{m-1}\,\alpha$ and due to (10)

$$\delta_{1m} \ge \frac{m}{m-1}\,\alpha\,. \tag{11}$$

Let $S_1 = \{I_1, \dots, I_k\}$ with $w_1 \ge w_2 \ge \cdots \ge w_k$ and $S_m = \{I'_1, \dots, I'_k\}$ with $w'_1 \ge w'_2 \ge \cdots \ge w'_k$. $w(S_1) > w(S_m)$ implies $w_1 > w'_k$ and therefore $w_1 \ge \delta_{1m}$.

We distinguish between two cases:

Case 1: $w_2 \ge \delta_{1m}$

Then $w(S_1) = 1 + \alpha \ge 2\delta_{1m} \ge \frac{2m}{m-1}\,\alpha$ and consequently $\alpha \le \frac{m-1}{m+1}$.

Suppose $\alpha = \frac{m-1}{m+1}$. Then $\Delta_{1i} = \delta_{1i} = \frac{m}{m-1}\,\alpha = \frac{m}{m+1}$ and $w(S_1) = \frac{2m}{m+1}$, $w(S_i) = \frac{m}{m+1}$, $i = 2, \dots, m$. Identifying items with their weights, this implies

$$S_1 = \left\{\frac{m}{m+1}, \frac{m}{m+1}, 0, \dots, 0\right\},$$

$$S_i = \left\{\frac{m}{m+1}, 0, \dots, 0\right\}, i = 2, \dots, m\,.$$

However, this partition is optimal. Therefore $\alpha < \frac{m-1}{m+1}$.

Case 2: $w_1 \ge \delta_{1m} > w_2$

Then $w_1 > w'_1 \ge w'_2 \ge \cdots \ge w'_k \ge w_2 \ge \cdots \ge w_k$.

There exists an item $I_s \in S_p$, $1 < p < m$, with $w_s < w_2$. Otherwise due to the lower bound $C^* \ge w_1 + w_2 + \cdots + w_k = w(S_1)$ (compare with (1)) the partition would be optimal. We conclude $m > 2$ and because of (10)

$$\Delta_{1p} \leq \delta_{1p} \leq w_2 - w_s \, .$$

Thus,

$$w(S_p) \geq w(S_1) - w_2 + w_s \, . \tag{12}$$

Because of (9)

$$\begin{aligned} w(S_p) &\leq m - \sum_{i \neq p} w(S_i) = m - w(S_1) - \sum_{i \neq 1,p} (w(S_1) - \Delta_{1i}) \\ &\leq m - (m-1)w(S_1) + \sum_{i \neq 1,p} \delta_{1i} \, . \end{aligned}$$

Using $\delta_{1i} \leq 1$ and $\delta_{1m} = w_1 - w_1'$ we get

$$w(S_p) \leq m - (m-1)w(S_1) + (m-3) + w_1 - w_1' \, . \tag{13}$$

Putting (12) and (13) together implies

$$w(S_1) - w_2 + w_s \leq 2m - 3 - (m-1)w(S_1) + w_1 - w_1' \, .$$

With $w_1 \leq 1$, $w_s \geq 0$ and $w_2 - w_1' \leq 0$

$$w(S_1) \leq 1 + \frac{m-2}{m} \, .$$

Finally, $w(S_1) = 1 + \alpha$ implies $\alpha \leq \frac{m-2}{m} < \frac{m-1}{m+1} = 1 - \frac{2}{m+1}$.

Now we show that for any $\varepsilon > 0$ there is a 2-optimal partition with relative error at least $1 - \frac{2}{m+1} - \varepsilon$.

For a given ε choose $\varepsilon' > 0$ and $p \in \mathbb{N}$ such that $\varepsilon' \leq \frac{\varepsilon}{m-1}$ and $p\varepsilon' = \frac{1}{m+1}$. Let $\frac{m}{m+1}, \frac{m}{m+1} - (m-1)\varepsilon', \varepsilon', \ldots, \varepsilon', 0, \ldots, 0$ be the weights of the items such that the sum of all weights is equal to m. Then

$$S_1^* = \left\{\frac{m}{m+1}, \varepsilon', \ldots, \varepsilon', 0, \ldots, 0\right\},$$

$$S_2^* = \left\{\frac{m}{m+1} - (m-1)\varepsilon', \varepsilon', \ldots, \varepsilon', 0, \ldots, 0\right\},$$

$$S_i^* = \{\varepsilon', \ldots, \varepsilon', 0, \ldots, 0\}, \quad i \geq 3,$$

is an optimal partition with $w(S_1^*) = \frac{m}{m+1} + p\varepsilon' = 1$, $w(S_2^*) = \frac{m}{m+1} - (m-1)\varepsilon' + (p+m-1)\varepsilon' = 1$ and $w(S_i^*) = (m+1)p\varepsilon' = 1$, $i \geq 3$.

On the other side, it is easy to verify that the partition

$$S_1 = \left\{\frac{m}{m+1}, \frac{m}{m+1} - (m-1)\varepsilon', 0, \ldots, 0\right\},$$

$$S_i = \{\varepsilon', \ldots, \varepsilon', 0, \ldots, 0\}, \quad i \geq 2,$$

with $w(S_1) = \frac{2m}{m+1} - (m-1)\varepsilon'$ and $w(S_i) = (mp+1)\varepsilon' = \frac{m}{m+1} + \varepsilon'$, $i \geq 2$, is 2-optimal.

The relative error is equal to $\frac{m-1}{m+1} - (m-1)\varepsilon' \geq 1 - \frac{2}{m+1} - \varepsilon$. ■

Unfortunately, the performance for the exchange algorithm is nearly the same as for the folding algorithm. Moreover, we could not find reasonable limits for the time complexity of this algorithm.

3.4 *A Primal-Dual Approach*

Usually, algorithms for multiprocessor scheduling belong to two main classes: The first are the "primal" algorithms with LPT as a typical representative. The number m of subsets S_i is fixed and we want to minimize the maximum weight of a subset.

"Dual" algorithms attack the problem the other way round. Each subset S_i is considered as a bin of a fixed capacity C and the objective is to minimize the number of bins used. Problems of this kind are usually called *bin packing* problems. Thus, a typical dual algorithm (for example Multifit [2]) applies a bin packing heuristic and embeds it in a binary search on the number of bins m.

Modifications of common dual algorithms like Multifit are not suitable for k-partitioning problems since they tend to put the "big" items in the same bin. Therefore we will construct a mixture of both methods. Due to this reason we prefer in this subsection mainly to speak of bins instead of subsets of a partition.

Recall that $level = \max\{a, b, c\}$ as defined in Lemma 2 is a lower bound for the optimum value of the k-partitioning problem.

In this subsection we will decribe our champion heuristic PD and we will show that it has a worst-case ratio of 4/3. PD will be introduced only for values $k \geq 4$. This is no disadvantage since for 3-partitioning problems heuristic MLPT is simpler and yields even a slightly better worst-case performance.

For each i ($i = m, m-1, \dots, 1$) heuristic PD calls the subprocedure PD(i) (dependent on parameter i) which assigns item sets $\{I'_1, \dots, I'_{ki}\}$ consisting of ki ($i = 1, \dots, m$) elements to i bins. Assume the corresponding weights w'_j to be sorted in nonincreasing order, i.e. $w'_1 \geq w'_2 \geq \cdots \geq w'_{ki}$.

Roughly spoken, subprocedure PD(i) checks for each bin B (with k_1 denoting the number of current items in B) whether the next consecutive $k - k_1$ items of the list can be put into B such that the total load of B does not exceed $\frac{4}{3}$*level*. In other words, PD tries to find a bin which it can fill with k items as much as possible (dual step). Only, if this is impossible for all bins, it will just apply LPT (primal step).

PD starts with procedure PD(m). By definition, PD(m) runs with the whole set of n items. After the call of procedure *PD(i)*, it is investigated whether all bins have total load not exceeding $\frac{4}{3}$*level*. If this is the case, PD stops with a partition of the elements. Otherwise, PD removes the first bin which was filled with k items by PD(i) (denoted by S_i). It can be shown that $w(S_i)$ does not exceed $\frac{4}{3}$*level*. Afterwards, the item set is reduced by the elements of S_i and PD calls subprocedure PD($i-1$).

In algorithmic notation PD(i) is defined in the following way:

Procedure PD(i):

(1) Initialize the bins $B_i^1, \dots, B_i^i$ to be empty and *open*. Set $j := 1$.

(2) Sort the open bins $B_i^1, \dots, B_i^i$ in nonincreasing order of weights, i.e. $w(B_i^1) \geq w(B_i^2) \geq \cdots \geq w(B_i^i)$. (If bins have equal weights, a bin with a smaller current number of elements is given a higher index. Otherwise, ties are broken arbitrarily.) Define the *filling weight*

$$\hat{w}(B_i^t) := w(B_i^t) + w'_j + w'_{j+1} + \cdots + w'_{j+k-|B_i^t|-1} \quad (t = 1, \dots, i) .$$

(3) If there is an index u such that $\hat{w}(B_i^u)$ does not exceed $\frac{4}{3}$*level*, take the smallest u and assign items $I'_j, I'_{j+1}, \dots, I'_{j+k-|B_i^u|-1}$ to bin B_i^u. *Close* bin B_i^u and set $j := j + k - |B_i^u|$. Goto 5.

(4) If there is no such index, assign item I_j^r to an open bin B_i^r with largest index r and set $j := j+1$. If $|B_i^r| = k$, close bin B_i^r.
(5) If $j > kl$ stop, else goto 2.

Now, heuristic PD reads as follows:

Heuristic PD:

(1) Set $i := m$, $L(m) := \{I_1, \dots, I_n\}$.
(2) Apply procedure PD(i) to item set $L(i)$. Call the produced partition $B_i(1), \dots, B_i(i)$.
(3) Sort the bins $B_i(1), \dots, B_i(i)$ in order of closing, i.e. $B_i(j)$ is closed before bin $B_i(j+1)$ $(j = 1, \dots, i-1)$.
(4) If $\max_{j=1,\dots,i} w(B_i(j)) \leq \frac{4}{3} level$, set $S_j := S_i(i-j+1)$ $(j = 1, \dots, i)$, output $S_1, \dots, S_m$ and stop.
(5) $S_i := B_i(1)$, $L(i-1) := L(i) \setminus S_i$, $i := i-1$.
(6) If $i = 1$ stop, else goto 2.

We can dispense with binary search in our algorithm, since in fact we show in the proof of Theorem 7 that the weight of the maximum set of the partition does not exceed $\frac{4}{3} level$. This is formulated in our main theorem.

Theorem 7: Let C^H denote the solution value obtained by procedure PD. Then we have

$$C^H \leq \frac{4}{3} level.$$

Proof: In the beginning we will introduce some notations which will be useful in the remainder of the proof. Recall that we denote the partition produced by procedure PD(i) and sorted in order of closing by $B_i(1), \dots, B_i(i)$. Note that $S_i = B_i(1)$. Further define the mean value of iteration i as

$$level(i) := \frac{1}{i}\left(\sum_{j=1}^{i} w(B_i(j))\right).$$

If a bin is closed in Step 3 of PD(i), we call it a *dual bin*, otherwise a *primal bin*. Analogously, we call an item a *dual item* if it is assigned to a bin in Step 3, otherwise a *primal item*. Thus, a dual bin is filled by assigning some next consecutive incoming items and has weight not exceeding $\frac{4}{3} level$. The dual items of that bin correspond to those items which have been used to fill the bin. If the items $w_{i_1},\dots,w_{i_k}$ of a dual bin are sorted in non-increasing order, i.e. $w_{i_1} \geq w_{i_2} \geq \cdots \geq w_{i_k}$, then there is an index $r \in \{0,\dots,k-1\}$ such that all items with index greater than r are dual items and the others are primal items, i.e. the primal items are always assigned first to a bin. Of course, in a primal bin only primal items exist. Note that primality or duality of an item or bin is varying with index i in heuristic PD.

The proof of Theorem 7 can be summarized as follows: It is sufficient to show that for each $i \in \{1,\dots,m\}$ the bin $B_i(1)$ has weight not exceeding $\frac{4}{3}$ *level*. The proof for this statement is split into several claims. Claim 2 and Claim 3 are used to show Claim 4. This claim says that if in some iteration step a bin B_1 with weight greater $\frac{4}{3}$ *level* arises then also a bin B_2 with weight smaller than *level* is produced. From Claim 1 it can be concluded that B_2 is closed before bin B_1. Therefore, bin B_1 is not identical to $B_i(1)$.

Claim 1: If there are r, s such that $w(B_i(r)) \leq$ level and $w(B_i(s)) > \frac{4}{3}$level, then $B_i(r)$ is closed before $B_i(s)$, i.e. $r < s$.

Proof: Let the items assigned to $B_i(s)$ be denoted by $p_1,\dots,p_k$ $(p_1 \geq p_2 \geq \cdots \geq p_k)$. Assume that bin $B_i(r)$ is closed after bin $B_i(s)$. Since $w(B_i(s)) > \frac{4}{3}$*level*, bin $B_i(s)$ is a primal bin. The assumption that $B_i(r)$ finishes later, implies $w(B_i(s)) - p_k \leq$ *level*. Therefore, $p_k > \frac{1}{3}$*level* and $k \leq 3$. We are finished since we defined PD only for $k \geq 4$. ■

Claim 2: The smallest item of S_i is larger than or equal to the smallest item of $S_{i-1} (i = 2,\dots,m)$.

Proof: If S_i and S_{i-1} are both produced in step 4 of PD, the claim follows from the fact that S_i and S_{i-1} are produced in the execution of subprocedure PD(t) for some $t \in \{1,\dots,m\}$ and that S_i is closed before S_{i-1}.

Therefore, w.l.o.g. $S_i = B_i(1)$ and $S_{i-1} = B_{i-1}(1)$ are the first bins to be closed in iteration i and $i-1$, respectively. Denote the items of list $L(i)$ by $p_1,\dots,p_{ki}$ $(p_1 \geq p_2 \geq \cdots \geq p_{ki})$. Let $p_{i(1)},\dots,p_{i(k)} (p_{i(1)} \geq p_{i(2)} \geq \cdots \geq p_{i(k)})$ be the elements contained in set S_i. Obviously, $L(i-1) = L(i) \backslash S_i$. Let $q_1,\dots,q_r$ and $\tilde{q}_{r+1},\dots,\tilde{q}_k (q_1 \geq q_2 \geq \cdots \geq q_r \geq \tilde{q}_{r+1} \geq \cdots \geq \tilde{q}_k)$ be the primal elements and the dual elements of S_{i-1}, respectively.

As long as only primal items are assigned to bins in iteration $i-1$, such a primal item I_v is also primal in iteration i since at that time the filling weight of any bin I_v in iteration i is not smaller than the filling weight of the corresponding bin in iteration $i-1$. Hence, the assignment of items to bins in iteration $i-1$ is the same as in iteration i until the first dual bin in iteration $i-1$ is produced. Recall that bin $S_i = B_{i-1}(1)$ is the first bin to be closed in iteration $i-1$.

In the case that $B_{i-1}(1)$ is a primal bin, there is a corresponding bin B' in iteration i which consists of the same items. Bin B' closes after bin $S_i = B_i(1)$ and also its smallest item is not greater than the smallest item of bin S_i.

Now consider the case that S_{i-1} is a dual bin. We have to show that $p_{i(k)} \geq \tilde{q}_k$. By definition,

$$w(S_{i-1}) \leq \frac{4}{3} level\,. \tag{14}$$

From the argumentation above we conclude that there is a bin in iteration i, denoted by $B_i(j)$, containing items $q_1, \ldots, q_r$. (If there are no primal items $q_1, \ldots, q_r$, take any bin which is empty just before assigning item q_1.) The remaining items of $B_i(j)$ shall be called $q_{r+1}, \ldots, q_k$ ($q_1 \geq q_2 \geq \cdots \geq q_k$). Note that $\tilde{q}_{r+1}$ is first dual item in iteration $i-1$. Consequently,

$$\tilde{q}_{r+1} \geq q_{r+1} \tag{15}$$

holds.

Let v be maximal with $p_v \in L(i-1)$ and $v < i(k)$, i.e. $p_v \geq p_{i(k)}$ and $p_v \notin S_i$. (If there is no such item, we have $S_i = \{p_1, \ldots, p_k\}$ and Claim 2 follows immediately.) Since S_i is the first bin to be closed in iteration i and q_k is the last item of bin $B_i(j)$, we obtain $q_k \leq p_{i(k)}$. Since p_v is not an element of S_i, but precedes some items of S_i in list $L(i)$, it is a primal item in iteration i. (Otherwise the bin containing p_v would be closed before S_i.)

Thus, procedure PD(i) checks whether p_v and some consecutive items fit into bin $B_i(j)$. We conclude that there are indices r', $v'(r' + v' - v + 1 = k)$ such that

$$q_1 + q_2 + \cdots + q_{r'} + p_v + p_{v+1} + \cdots + p_{v'} > \frac{4}{3}\, level\,. \tag{16}$$

$\tilde{q}_{r+1}, \ldots, \tilde{q}_k$ are dual items in iteration $i-1$. Consequently, they are consecutive in list $L(i-1)$. All other items between $\tilde{q}_{r+1}$ and $\tilde{q}_k$ in list $L(i)$ are assigned to bin S_i. Together with (15) it can be concluded for $r' > r$

$$q_1 + q_2 + \cdots + q_{r'} \leq q_1 + q_2 + \cdots + q_r + \tilde{q}_{r+1} + \tilde{q}_{r+2} + \cdots + \tilde{q}_{r'}\,. \tag{17}$$

Combining (14), (16) and (17) we get

$$\tilde{q}_{r'+1} + \tilde{q}_{r'+2} + \cdots + \tilde{q}_k < p_v + p_{v+1} + \cdots + p_{v'}\,.$$

Both sides of this inequality consist of the same number of terms. Therefore,

$p_v > \tilde{q}_k$ and $p_{i(k)} \geq \tilde{q}_k$ due to the minimality of p_v. (For $r' \leq r$ the proof is analogous.) ■

Claim 3: Let $\theta(i)$ be the elements of list $L(i)(i = 1, \ldots, m)$ with size greater than level/3. Denote the LPT-partition of $\theta(i)$ in i sets by $T_1(i), \ldots, T_i(i)$. W.l.o.g. assume that $w(T_1(i)) \geq w(T_2(i)) \geq \cdots \geq w(T_i(i))$. Set $\lambda_\ell(i) := \sum_{j=1}^{\ell} |T_j(i)|$ for $\ell = 1, \ldots, i$. Then the following two properties hold:

(i) *LPT applied to $\theta(i)$ on i bins does not assign three elements of $\theta(i)$ to one bin.*
(ii)

$$level \geq \max_{\ell=1,\ldots,i} \frac{1}{\ell} \left(\sum_{j=1}^{\ell} w(T_j(i)) + \sum_{j=1}^{\ell k - \lambda_\ell(i)} w_{n-j+1} \right)$$

Proof: First note that we extend the definition of LPT analogously to Steps 2 and 4 in *PD*(ℓ) in the following way: If there are several bins with smallest total weight, LPT shall assign the current item to one of the bins which contain the smallest number of elements. W.l.o.g. we may assume that $\theta(i)$ is nonempty.

The proofs of (i) and (ii) will be done simultaneously by induction on iteration step i.

$i = m$: By the definition of p in Section 2 we obtain that $w_{p+1} \leq level/3$, otherwise (6) would contradict $c \leq level$. Therefore, $\theta(m) \subseteq \{I_1, \ldots, I_p\}$, and no more than two items of $\theta(m)$ are assigned to the same bin by LPT. (i) follows.

Denote by $\gamma := \lambda_m(m)$ the number of items greater than $level/3$. If $\gamma \leq m$, (ii) can be concluded from (3). Otherwise, $\gamma \leq p \leq 2m$. Since $w_{\gamma-2} + w_{\gamma-1} + w_\gamma > level$, we get $b(\gamma) = \hat{b}(\gamma)$ and (ii) is a consequence of (4) and (6).

$i + 1 \to i$: For $\lambda_i(i) < m$, (i) is obvious. So, assume $\lambda_i(i) \geq m$. Let p_1, p_2 be the two largest elements of $S_{i+1} (p_1 \geq p_2)$. Since S_{i+1} is the first set to be closed in iteration $i + 1$, item p_1 is greater than $level/3$, i.e. item p_1 is the largest element of $\theta(i + 1)$ which is cut from $L(i + 1)$ in iteration $i + 1$. Thus, there is a set $T \in \{T_1(i + 1), \ldots, T_{i+1}(i + 1)\}$ with $p_1 \in T$. By induction hypothesis T contains at most one further item p'. As long as only primal items are put into bins in step $i + 1$, the assignment of the elements of $\theta(i + 1)$ is the same as in the LPT-partition of $\theta(i + 1)$. Obviously, $p_2 = p'$, if S_{i+1} is a primal bin. If S_{i+1} is a dual bin, even $p_2 \geq p'$ holds. Summarizing, we can say that $\theta(i + 1) \setminus T$ *dominates* $\theta(i)$, i.e. for each item $q' \in \theta(i)$ there is a $q'' \in \theta(i + 1) \setminus T$ such that $q'' \geq q'$, (All q'' shall be different elements.)

Let $q_1, \ldots, q_r$ be the elements of $\theta(i)(q_1 \geq q_2 \geq \cdots \geq q_r)$. Assume that (i) does not hold for iteration i and let bin $B_i(s)(1 \leq s \leq i)$ be the first bin, to which LPT assigns three elements of $\theta(i)$. Denote these elements by q_s, q_{2i-s+1} and $q_t(s \leq 2i - s + 1 \leq t)$ and denote by A the LPT assignment of $\{q_1, \ldots, q_t\}$.

By our special definition of LPT all items which are alone in a bin in assignment A, have weight exceeding $q_s + q_{2i-s+1}$.

Domination, the induction hypothesis and the pidgeon hole principle imply that there is at least one such a single element $q_{r'}(r' \leq r)$, which is paired with an item $q_{t'}$ with $q_{t'} \geq q_t$ in the LPT-partition of $\theta(i+1)$. We conclude

$$q_s + q_{2i-s+1} < q_{r'}$$

and by induction hypothesis (ii) we have for $\ell = 1$

$$level \geq q_{r'} + q_{t'} > q_s + q_{2i-s+1} + q_t \ .$$

This contradicts $q_t \in \theta(i)$ and (i) follows.
Domination and (i) have the effect that

$$\sum_{j=1}^{\ell} w(T_j(i+1)) \geq \sum_{j=1}^{\ell} w(T_j(i))$$

holds for each $\ell \in \{1, \ldots, i\}$. We get (ii) as direct result from our induction hypothesis. ■

The next claim will be shown by induction on $i (i = m, m-1, \ldots, 1)$, as well.

Claim 4: $level(i) \leq level$.

Proof: $i = m$: $level(m) \leq a \leq level$ is obvious.
$i + 1 \to i$: Assume that $level(i+1) \leq level$ holds. If $w(S_{i+1}) \geq level$, then $level(i) \leq level(i+1) \leq level$ and we are done. Therefore, assume $w(S_{i+1}) < level$.

Let $p_1, \ldots, p_{k(i+1)} (p_1 \geq p_2 \geq \cdots \geq p_{k(i+1)})$ denote the items in list $L(i+1)$ and $v_1, \ldots, v_k (v_1 \geq v_2 \geq \cdots \geq v_k)$ the items of S_{i+1}, respectively.

Consider first the case that S_{i+1} is a primal bin.. Let $p_d = v_k$. Since S_{i+1} is primal and the first bin to be closed in iteration $i+1$, $p_{d-1} \neq v_{k-1}$ holds and

$$v_1 + v_2 + \cdots + v_{k-1} + p_{d-1} > \frac{4}{3} level \ .$$

Therefore, $v_{k-1} \geq p_{d-1} - v_k > \frac{1}{3} level$. It follows $k \leq 3$, a contradiction to the assumption $k \geq 4$.

Consider now the case that S_{i+1} is a dual bin. Denote the primal items of S_{i+1} by $v_1, \ldots, v_r$ and the dual items by $p_s, p_{s+1}, \ldots, p_t (r + t + 1 - s = k)$. By assumption,

$$v_1 + v_2 + \cdots + v_r + p_s + p_{s+1} + \cdots + p_t < level\ .$$

Obviously, $p_{s-1} \neq v_r$, and we conclude

$$v_1 + v_2 + \cdots + v_r + p_{s-1} + p_s + \cdots + p_{t-1} > \frac{4}{3} level\ .$$

Combining these two inequalities, we get

$$p_{s-1} > \frac{1}{3} level\ . \tag{18}$$

Furthermore, all items in $P := \{p_1, \ldots, p_{s-1}\}$ are primal and are assigned to the bins by LPT-rule. Let $q_1, \ldots, q_k$ $(q_1 \geq q_2 \geq \cdots \geq q_k)$ be the items of any bin $B_{i+1}(j)$ with $j \in \{2, \ldots, i+1\}$. Let $q_1, \ldots, q_u$ be the elements assigned to $B_{i+1}(j)$ before item p_s. The items $q_1, \ldots, q_u$ are primal and by (18) we obtain $q_u > \frac{1}{3} level$. Since $\{q_1, \ldots, q_u\} \subseteq P$, Claim 3(i) implies that $u \leq 2$.

Thus, all items in the set $Q(j) := \{q_{u+1}, \ldots, q_k\}$ are assigned to bin $B_{i+1}(j)$ after p_t and are not larger than this element. Let $p_{t+1} = w_v$ in the original sorting of the items. We get

$$\bigcup_{j=2}^{i+1} Q(j) \subseteq \{w_v, \ldots, w_n\}\ .$$

From Claim 2 it can even be deduced that

$$\bigcup_{j=2}^{i+1} Q(j) = \{w_v, \ldots, w_n\}\ .$$

But now it is a direct consequence from Claim 3(ii) that the average weight of items in the bins $\{B_{i+1}(j) \mid j = 2, \ldots, i+1\}$ does not exceed *level*. Clearly, $level(i) \leq level$. ■

We have now enough prerequisites to finish the proof of our main theorem. In the case that in some iteration step i a bin $B_i(s)$ with $w(B_i(s)) > \frac{4}{3} \mathit{level}$ appears, Claim 4 shows that there is also a bin $B_i(r)$ with $w(B_i(r)) < \mathit{level}$. But Claim 1 implies that $w(S_i) = w(B_i(1)) \leq \frac{4}{3} \mathit{level}$ for each $i \in \{1, \ldots, m\}$.

4 Conclusions and Open Problems

In this paper we presented several heuristics for the k-partitioning problem. The folding algorithm and the exchange algorithm have a worst-case ratio which is close to 2. The major contribution is the heuristic PD with a performance ratio of $\frac{4}{3}$. We finish with a collection of open problems:

(1) The exchange algorithm is of no theoretical and practical interest as long as there are no results about its time complexity. Hence, it is interesting to decide whether the exchange algorithm requires polynomial time or to construct a variation of this heuristic which is a polynomial algorithm.
(2) We do not know anything about the tightness of the $\frac{4}{3}$ bound for the algorithm PD. From the construction of the lower bound we conjecture that the bound is tight. By exchanging *level* with a general lower bound ℓ for the optimum solution and approximating ℓ by binary search, it seems possible that a better worst-case ratio can be obtained.
(3) There is no evidence that the solution designed by procedure PD(m) has a performance ratio worse than $\frac{4}{3}$.
(4) Even there is a chance that a simple modified version of the LPT, where we just close a bin when k items are assigned to it, yields still a worst-case ratio of $\frac{4}{3}$.
(5) The most interesting open problem is to find a reasonable approximation algorithm for the following generalization of the k-partitioning problem. It is called the k_i *– partitioning problem*: We are given n items and integers $k_1, \ldots, k_n$ with $n \leq \sum_{i=1}^{n} k_i$. The objective is to find a partition into subsets $S_1, \ldots, S_n$ such that each set $S_i (i = 1, \ldots, n)$ contains at most k_i items and the maximum weight of all subsets is as small as possible.

References

[1] Chen B (1993) A note on LPT scheduling. Op. Res. Letters 14:139–142
[2] Coffman EG Jr, Garey MR, Johnson DS(1978) An application of bin-packing to multiprocessor scheduling. SIAM J. Comput. 7:1–17

[3] Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco

[4] Graham RL (1966) Bounds for certain multiprocessing anomalies. Bell System Tech. 45:1563–1581

[5] Graham RL (1969) Bounds on multiprocessing timing anomalies. SIAM J. Appl. Math. 17:263–269

[6] Kellerer H, Kotov V, A 7/6 approximation algorithm for 3-partitioning and its application to multiprocessor scheduling. Submitted for publication

[7] Kellerer H, Woeginger G (1993) A tight bound for 3-partitioning. Discr. Appl. Math. 45:249–259

[8] EL Lawler EL Lenstra JK, Rinnoy Kan AHG, Shmoys DB (1993) Sequencing and scheduling: Algorithms and complexity. In: Handbook of Operations Research, volume 4, North-Holland, Amsterdam, pp. 445–552

Received: August 1995
Revised version received: July 1997