On Packing Splittable Items With Cardinality Constraints

Fouad B. Chedid

Department of Computer Science, Notre Dame University
P.O.Box: 72 Zouk Mikael, Zouk Mosbeh, Lebanon
fchedid@ndu.edu.lb

Abstract. This paper continues the study of the the allocation of memory to processors in a pipeline problem. This problem can be modeled as a variation of bin packing where each item corresponds to a different type and the normalized weight of each item can be greater than 1, which is the size of a bin. Furthermore, in this problem, items may be split arbitrarily, but each bin may contain at most k types of items, for any fixed integer $k \geq 2$. The case of $k = 2$ was first introduced by Chung *el al.* who gave a 3/2-approximation asymptotically. In this paper, we generalize the result of Chung *et al.* to higher k. We show that NEXT FIT gives a $\left(1 + \frac{1}{k}\right)$-approximation asymptotically, for $k \geq 2$. Also, as a minor contribution, we rewrite the strong NP-hardness proof of Epstein and van Stee for this problem for $k \geq 3$.

1 Introduction

An important issue in parallel processing is the allocation of memory to processors. In principle, each processor should have enough memory with little memory waste. In 2006, Chung *et al.* [1] studied this problem in the context of designing fast IP lookup schemes where the processors are arranged as a pipeline. Knowing that most existing IP lookup schemes traverse some kind of a tree [6] (lookup time is proportional to the height of the tree), a simple way to statically pipeline a tree is to place all nodes at height i in memory unit i which is then made accessible only to processor numbered i. Obviously, while this design prevents memory contention, it is not very efficient in terms of memory utilization. The question at hand then is how to assign memory to processors so that both memory contention and memory waste are minimized. According to [1], this problem was first raised and left as an open problem in [7]. To deal with this problem, the authors of [1] proposed to allocate memory dynamically rather than statically. Consequently, they proposed an architecture that connects processors to multiple two-port memories using a crossbar switch interconnection network [2]. This allows each processor to be connected to multiple memories, but allows at most two processors to be connected to a single memory. As observed in [1], the crossbar needs only be configured at allocation time, which is generally orders of magnitude less stringent than lookup times. The formulation of the problem

as proposed by Chung *et al.* gives the following definition, which we term the two-port memory allocation problem:

The Two-Port Memory Allocation Problem:
INPUT: A number of processors n, a number of memories m, and a collection of memory requests per processor.
OUTPUT: A way to satisfy each processor's request such that no more than two processors are allocated to any one memory.

The authors of [1] abstracted this problem as a variant of bin packing, where the bins are the memories and the items to be packed are the memory requirements of the processors, where each processor corresponds to a different type. Thus, the two-port constraint is abstracted as a two-type constraint. Moreover, since processors may require memory of any size (its normalized value can be larger than 1, which is the size of a bin), this version of bin packing allows items to be split arbitrarily, but each bin may contain at most two types of items. The authors of [1] showed that this problem is NP-hard in the strong sense. They used a reduction from the 3-Partition problem [4]. They also gave a $O(n)$ time 3/2-approximation asymptotically. Moreover, this approximation is optimal if $m > n$.

In 2007, Epstein and van Stee proposed a generalization of the solution proposed by Chung *et al.* In particular, they proposed to allow each memory to be accessed by at most k processors, for any fixed integer $k \geq 2$. We term this generalization the k-port Memory Allocation Problem, which can be similarly defined.

The k-Port Memory Allocation Problem:
INPUT: A number of processors n, a number of memories m, and a collection of memory requests per processor.
OUTPUT: A way to satisfy each processor's request such that no more than k processors are allocated to any one memory. Here, k is any fixed integer greater than or equal to 2.

Modeled as a bin packing problem, this generalization still allows items to be split arbitrarily, but now each bin may contain at most k types of items for any fixed integer $k \geq 2$. The authors of [3] showed that this generalization is also NP-hard in the strong sense. They used a reduction from the 3-Partition problem. They also showed that a straightforward generalization of NEXT FIT gives a $\left(2 - \frac{1}{k}\right)$-approximation.

In this paper, we generalize the approximation result of Chung *et al.* to the k-Port Memory Allocation Problem. We show that NEXT FIT gives a $\left(1 + \frac{1}{k}\right)$-approximation asymptotically, for $k \geq 2$. Also, we rewrite the NP-hardness proof of Epstein and van Stee for this problem for $k \geq 3$.

The rest of the paper is organized as follows. Section 2 reviews basic definitions and relevant results from the literature. Our contributions are included in sections 3. Section 4 is the conclusion.

2 Preliminaries

Let P be a minimization problem and let I be an instance of P. Let A be an algorithm for P and let $A(I)$ be the cost of algorithm A on the input I. Let $OPT(I)$ be the cost of an optimal algorithm for P on the input I. We define the approximation ratio $R_A(I)$ by

$$R_A(I) = \frac{A(I)}{OPT(I)}.$$

The absolute approximation ratio R_A for the algorithm A for P is given by [4]

$$R_A = \inf\{r \geq 1 : R_A(I) \leq r, \forall I\}.$$

Thus, for all inputs I of P, $A(I) \leq R_A \cdot OPT(I)$. In this case, the algorithm A is called an R_A-approximation for P. This means that the algorithm A guarantees a solution for P that is within a factor of R_A of the optimum.

We next give the definitions of five problems:

The Partition Problem Given a set of items of total size B, the partition problem asks for a way to partition these items into two subsets of size $B/2$. This problem is known to be NP-hard [4].

The 3-Partition Problem Given a set of $3m$ positive numbers $s_1, s_2, \ldots, s_{3m}$ such that $\sum_{j=1}^{3m} s_j = mB$ and each s_i satisfies $B/4 < s_i < B/2$, the 3-partition problem asks for a way to partition the $3m$ numbers into m sets of size 3 such that the sum of the elements of each set is exactly B. This problem is known to be NP-hard in the strong sense [4].

The Classical Bin Packing Problem Given an infinite number of bins each of capacity 1, and a list of items of weights $\{w_1, w_2, \ldots, w_n\}$, where $w_i \in (0, 1]$, the classical bin packing problem asks for a way to pack these items into a minimum number of bins. A simple reduction from the 3-partition problem shows that bin packing is NP-hard. This reduction also shows that no polynomial-time algorithm for bin packing can have an approximation ratio better than $3/2$ unless P=NP [4]. A simple online 2-approximation for bin packing is NEXT FIT [5]. Recall that in the NEXT FIT heuristic, an item is placed in the current bin if it fits. If the item does not fit, the current bin is closed, and another bin is considered.

The 2-Way Splittable Bin Packing Problem In [1], the authors introduced a variation of bin packing, where each item corresponds to a different type and the normalized weight of each item can be greater than 1, which is the size of a bin. Furthermore, in this problem, items may be split arbitrarily, but each bin may contain at most two types of items. Hereafter, we refer to this problem as

the 2-way Splittable Bin Packing problem (2-SBP for short). The authors of [1] showed that 2-SBP is NP-hard in the strong sense. They used a reduction from the 3-partition problem. They also gave a 3/2-approximation, asymptotically, for 2-SBP. In particular, the approximation ratio of Chung *et al.*'s algorithm for 2-SBP is $\frac{3}{2}(1 + o(1))$ as the sum of weights of the items tends to infinity.

The *k*-Way Splittable Bin Packing Problem In [3], the authors introduced a generalization of 2-SBP, where items can still be split arbitrarily, but each bin may contain at most k types of items, for any fixed integer $k \geq 2$. We refer to this problem as k-SBP. Epstein and van Stee [3] showed that k-SBP is NP-hard in the strong sense. They used a reduction from the 3-partition problem. They also showed that a straightforward generalization of NEXT FIT gives a $(2 - 1/k)$-approximation for k-SBP.

3 Our Contribution

3.1 A Generalization of the Result of Chung *et al.*

The following generalization of NEXT FIT for k-SBP is quoted from [3]: An item is packed (partially) in the current bin if the bin is not full and the bin contains less than k types of items so far. If the item does not fill entirely in the current bin, the current bin is filled, closed, and as many new bins are opened as necessary to contain the item.

We prove the following theorem:

Theorem 1. *The approximation ratio of NEXT FIT for k-SBP is $1+\frac{1}{k}$ asymptotically.*

Proof. This proof is inspired by the proof of Theorem 2 from [1]. Let the sizes of the items to be packed be $W = \{w_1, w_2, \ldots, w_n\}$ with the understanding that w_i is the size of the item of type i $(1 \leq i \leq n)$. Recall that each item corresponds to a different type. Let $w = \sum_{i=1}^{n} w_i$ and $w^* = \sum_{i=1}^{n} \lceil w_i \rceil$. Let NF and OPT denote the number of bins needed in NEXT FIT and the optimum packing, respectively. We develop our proof in three steps.

Lemma 1. *For $k \geq 2$,*

$$OPT \geq \max\{w, w^*/k\}.$$

Proof. Clealy, $OPT \geq w$. It remains to show that $OPT \geq w^*/k$. For each item i, there are at least $\lceil w_i \rceil$ parts of item i. Since, there can be at most k parts of items per bin, it follows that

$$OPT \geq \frac{1}{k} \sum_{i=1}^{n} \lceil w_i \rceil = w^*/k.$$

Lemma 2. *For $k \geq 2$,*

$$\frac{w + w^* + k - 1}{k} \leq \left(1 + \frac{1}{k}\right)(1 + o(1)) \max\{w, w^*/k\}$$

as $w \to \infty$.

Proof. We consider two cases:
Case a: If $w \leq w^*/k$, then $\frac{w+w^*+k-1}{k} \leq \frac{(1+k)w^*+k^2-k}{k^2} = \left(1 + \frac{1}{k}\right)\left(1 + \frac{k(k-1)}{(k+1)w^*}\right)\frac{w^*}{k}$ $= \left(1 + \frac{1}{k}\right)(1 + o(1))\frac{w^*}{k}$, as $w \to \infty$.
Case b: If $w^*/k < w$, then $\frac{w+w^*+k-1}{k} < \frac{(k+1)w+k-1}{k} = \left(1 + \frac{1}{k}\right)\left(1 + \frac{k-1}{(k+1)w}\right)w =$ $\left(1 + \frac{1}{k}\right)(1 + o(1))w$, as $w \to \infty$.

Lemma 3. *For any $k \geq 2$,*

$$NF \leq \frac{w + w^* + k - 1}{k}.$$

Proof. First, we show that we may assume that each bin completely filled by NEXT FIT contains exactly k types of items for any $k \geq 2$.

Our proof proceeds by induction on NF. Suppose that NEXT FIT has a bin B which is completely filled with at most α types of items for $1 \leq \alpha \leq k-1$. Denote these types of items by $t_1, t_2, \ldots, t_\alpha$. W.l.o.g., assume that these types appear in B in the order $t_1, t_2, \ldots, t_\alpha$ with the (part of) item of type t_1 at the bottom of B and the (part of) item of type t_α on top of B. Observe that items of types $t_2, \ldots, t_{\alpha-1}$ are not split by NEXT FIT. Let s_1 and s_α denote the sizes of the (parts of) items of types t_1 and t_α in B, respectively. Let W' denote a variation of W in which the items of types $t_2, \ldots, t_{\alpha-1}$ are removed, and the items of types t_1 and t_α have their sizes reduced by s_1 and s_α, respectively. Let NF' denote the number of bins needed in NEXT FIT to pack W'. Let $w' = \sum_{i \in W'} w_i$ and $w'^* = \sum_{i \in W'} \lceil w_i \rceil$. Recall that $w = \sum_{i \in W} w_i$ and $w^* = \sum_{i \in W} \lceil w_i \rceil$. We have

$$w' = w - \left(\sum_{j=2}^{\alpha-1} w_{t_j}\right) - s_1 - s_\alpha = w - 1.$$

This is true because $\sum_{j=2}^{\alpha-1} w_{t_j} + s_1 + s_\alpha = 1$.

Also, we have

$$w'^* \leq w^* - \sum_{j=2}^{\alpha-1} \lceil w_{t_j} \rceil \leq w^* - (\alpha - 2) \leq w^* + 1.$$

In this equation, we used the fact that $\lceil w'_{t_1} \rceil \leq \lceil w_{t_1} \rceil$, since $0 < s_1 < 1$ (and similarly for w'_{t_α}) and the fact that $\alpha \geq 1$.

Now, since the packing of instance W' uses a smaller number of bins ($NF' = NF - 1$), by the induction hypothesis, we have

$$NF' \leq \frac{w' + w'^* + k - 1}{k}.$$

Since $NF' = NF - 1$, $w' = w - 1$ and $w'^* \leq w^* + 1$, we have

$$NF \leq \frac{w + w^* + k - 1}{k}.$$

This completes the proof that each bin completely filled by NEXT FIT may be assumed to contain exactly k types of items.

Next, and following [3], we define a block as a maximal set of bins which were consecutively filled by NEXT FIT in which each pair of consecutive bins contains parts of the same item. Denote the list of blocks by $\{B_1, B_2, \ldots, B_m\}$, and let b_i denote the number of bins in block B_i ($1 \leq i \leq m$). As observed in [3], in each block, all bins are completely filled except perhaps for the last bin, which contains k (parts of) items (except perhaps for block B_m). For a given block B_j ($1 \leq j \leq m$), we use the notation $\sum_{i \in B_j} w_i$ to indicate the sum of weights of the items packed in bins pertaining to block B_j. Let $w_{B_j} = \sum_{i \in B_j} w_i$ and $w^*_{B_j} = \sum_{i \in B_j} \lceil w_i \rceil$. To prove Lemma 3, it is enough to show that the number of bins b_j in block B_j is at most $\frac{w_{B_j} + w^*_{B_j} + k - 1}{k}$, for $1 \leq j \leq m$.

First, we consider blocks B_j, for $1 \leq j \leq m-1$. Here, we consider two cases. First, we consider the case where the last bin of B_j is full. In this case,

$$b_j = w_{B_j} \tag{1}$$

Let n_j denote the total number of items in bins $1, \ldots, b_j$ of block B_j. Then

$$w^*_{B_j} = \sum_{i \in B_j} \lceil w_i \rceil \geq n_j.$$

Since we may assume that each bin in B_j contains exactly k types of items, we have the following about block B_j:

- There are exactly $k - 1$ unsplit items in each of the bins numbered 1 and b_j of B_j.
- There are exactly $k-2$ unsplit items in each of the bins numbered $2, \ldots, (b_j - 1)$ of B_j.
- There are exactly $b_j - 1$ items whose parts extend bin i into bin $(i + 1)$ of B_j for $1 \leq i \leq b_j - 1$.

Thus, the total number of items in B_j is

$$n_j = 2(k-1) + \sum_{i=2}^{b_j - 1} (k-2) + b_j - 1.$$

Thus

$$w^*_{B_j} \geq n_j = kb_j - b_j + 1 \tag{2}$$

Equations 1 and 2 imply

$$w_{B_j} + w^*_{B_j} \geq b_j + kb_j - b_j + 1 = kb_j + 1.$$

Thus

$$b_j \leq \frac{w_{B_j} + w^*_{B_j} - 1}{k} \leq \frac{w_{B_j} + w^*_{B_j} + k - 1}{k}.$$

Next, we consider the case where the last bin of B_j, which contains k types of items, is partially filled. In this case, we have

$$w_{B_j} \geq b_j - 1 \tag{3}$$

This is true since there are at least $b_j - 1$ filled bins in B_j. Equations 2 and 3 imply

$$w_{B_j} + w^*_{B_j} \geq b_j - 1 + kb_j - b_j + 1 = kb_j.$$

Thus

$$b_j \leq \frac{w_{B_j} + w^*_{B_j}}{k} \leq \frac{w_{B_j} + w^*_{B_j} + k - 1}{k}.$$

This is true because $k \geq 2$. This completes the proof for all blocks B_j, for $1 \leq j \leq m - 1$.

Next, we consider the case of block B_m. The last bin of this block may not contain any unsplit items at all. That is, the last bin of B_m may contain only a part of item that was extended into it from the previous bin. In this case, the total number of items in block B_m is

$$n_m \geq kb_m - b_m + 1 - (k - 1).$$

Thus

$$w^*_{B_m} \geq n_m = kb_m - b_m - k + 2 \tag{4}$$

Equations 1 and 4 imply

$$w_{B_m} + w^*_{B_m} \geq b_m - 1 + kb_m - b_m - k + 2 = kb_m - k + 1.$$

Thus

$$b_m \leq \frac{w_{B_j} + w^*_{B_j} + k - 1}{k}.$$

This completes the proof of Lemma 3.

Finally, putting Lemmas 1, 2, and 3 together gives, as $w \to \infty$

$$NF \leq \frac{w + w^* + k - 1}{k} \leq \left(1 + \frac{1}{k}\right)(1+o(1)) \max\{w, w^*/k\} \leq \left(1 + \frac{1}{k}\right)(1+o(1)) \cdot OPT.$$

This completes the proof of Theorem 1.

3.2 The NP-Hardness Proof of k-SBP Revisited

The Original Proof: The following theorem appears in [3]:

Theorem 2. *Packing splittale items with a cardinality constraint of k parts of items per bin is NP-hard in the strong sense for any fixed $k \geq 3$.*

Proof. Given an instance of 3-partition and a fixed $k \geq 3$, an instance of k-SBP is constructed as follows. There are $m(k-3)$ items, called padding items, all of size $\frac{3k-1}{3k(k-3)}$ (for $k = 3$, no items are defined at this point). In addition, there are $3m$ items, called adapted items, where item j has size $s_j/(3kB)$ (for $k = 3$, the size is defined to be s_j/B). The goal is to find a packing with exactly m bins. Since there are mk items, a solution in m bins contains exactly k items per bin. Since the sum of items is exactly m, all bins in such a solution are completely filled.

(*only if*) If there exists a partition of the numbers into m sets of sum B each, then there is a partition of the adapted items into m sets of sum $1/(3k)$ each (the sum is 1 for $k = 3$). Each bin is packed with $k-3$ padding items and one such triple of adapted items, giving m sets of k items, each set of sum exactly 1.

(*if*) If there is a packing into exactly m bins, no items are split and each bin must contain exactly k items. It is shown that for $k \geq 4$ each bin contains exactly $k-3$ padding items, and therefore contains exactly 3 adapted items, whose total size is exactly $1/(3k)$ (the sum is 1 for $k = 3$). These three adapted items correspond to three numbers in the instance of the 3-partition problem whose sum is exactly B. Thus, a solution in m bins implies a partition.

Our Version of the Proof: Given an instance of the 3-partition problem and a fixed $k \geq 3$, we define an instance of k-SBP as follows. There are $m(k-3)$ items, called padding items, all of size $1/k$. In addition, there are $3m$ items, called adapted items, where item j has size $3s_j/(kB)$. The goal is to find a packing with exactly m bins. Since there are mk items, a solution in m bins contains exactly k items per bin. Since the sum of items is exactly m $(= m(k-3) \cdot (1/k) + 3/(kB) \cdot mB)$, all bins in such a solution are completely filled. Next, we show that there is a partition if and only if there is a solution in m completely occupied bins.

(*only if*) If there exists a partition of the numbers into m sets of sum B each, then there is a partition of the adapted items into m sets of sum $3/k$ each. Each bin is packed with $k-3$ padding items and one such triple of adapted items, giving m sets of k items, each set of sum exactly 1 $(= (k-3) \cdot (1/k) + 3/k)$.

(*if*) If there is a packing into exactly m bins, no items are split and each bin must contain exactly k items. If $k = 3$, then the k-SBP instance has no padding items. Each bin contains exactly 3 adapted items, whose total size is exactly 1. These three adapted items correspond to three numbers in the instance of the 3-partition problem whose sum is exactly B. Thus, a solution in m bins implies a partition. Consider the case of $k \geq 4$. First, we prove that each bin contains

exactly $k-3$ padding items. For $1 \leq i \leq m$, let x_a^i and x_p^i be the numbers of adapted and padding items in the i^{th} bin, respectively. Moreover, for $1 \leq j \leq x_a^i$, let a_{ij} be the size of the j^{th} adapted item in the i^{th} bin. By construction, we have the following, for all $1 \leq i \leq m$:

$$x_a^i + x_p^i = k \tag{5}$$

$$\sum_{j=1}^{x_a^i} a_{ij} + \frac{x_p^i}{k} = 1 \tag{6}$$

$$\frac{3x_a^i}{4k} < \sum_{j=1}^{x_a^i} a_{ij} < \frac{3x_a^i}{2k} \tag{7}$$

Equation 5 states that the total number of items per bin is exactly k. Equation 6 states that the total size of adapted and padding items per bin is exactly 1. Equation 7 enforces proper bounds on the total size of adapted items per bin. These bounds are due to the bounds defined on the numbers in the instance of the 3-partition problem. Solving equations 5, 6, and 7 gives $x_a^i \leq 2k/3$ and $x_p^i \geq k/3 \geq 1$, since $k \geq 4$. Thus, for $k \geq 4$, each bin must contain at least 1 padding item. Given this information, the problem now reduces to packing $3m$ adapted items and $m(k-4)$ padding items into exactly m bins where each bin contains exactly $k-1$ items whose total size is exactly $1-1/k$. Repeating the above calculations on this new instance gives $x_a^i \leq 2(k-1)/3$ and $x_p^i \geq (k-1)/3 \geq 1$, since $k-1 \geq 4$. Thus, each bin in this new instance must contain at least 1 padding item. Repeating this argument again and again (as long as the exact number of items to be packed per bin is ≥ 4) distributes the $m(k-3)$ padding items of the original instance evenly across the m bins. Thus, each bin contains exactly $k-3$ padding items. Equivalently, each bin contains exactly 3 adapted items, whose total size is exactly $1 - \frac{k-3}{k} = 3/k$. These three adapted items correspond to three numbers in the instance of the 3-partition problem whose sum is exactly B. Thus, a solution in m bins implies a partition.

4 Conclusion

In this paper, we continued the study of the the allocation of memory to processors in a pipeline problem. This problem is modeled as a variant of bin packing named k-way splittable bin packing (k-SBP for short). In k-SBP, each item corresponds to a different type and items may be split arbitrarily but each bin may contain at most k types of items, for any fixed integer $k \geq 2$. We generalized the result of Chung *et al.* for 2-SBP to k-SBP. In particular, we showed that a straightforward generalization of NEXT FIT gives a $\left(1 + \frac{1}{k}\right)$-approximation asymptotically. Also, we rewrote the NP-hardness proof of Epstein and van Stee for k-SBP for $k \geq 3$.

References

1. Chung, F., Graham, R., Mao, J., Varghese, G.: Parallelism Versus Memory Allocation in Pipelined Router Forwarding Engines. Theory of Computing Systems **39:6** (2006) 829–849
2. Culler, D., Singh, J., Gupta, A.: Parallel Computer Architecture, A Hardware/Software Approach. Morgan Kaufman, San Mateo, CA, 1999
3. Epstein, L., van Stee, R.: Improved Results for a Memory Allocation Problem. In: Algorithms and Data Structures. Lecture Notes in Computer Science, Vol. 4619. Springer-Verlag, Berlin Heidelberg (2007) 362-373
4. Garey, M. R., Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York, 1979
5. Johnson, D. S.: Fast Algorithms for Bin Packing. Journal of Computer and System Sciences **8:3** (1974) 272-314
6. Ruiz-Sanchez, M., Biersack, E., Dabbous, W.: Survey and Taxonomy of IP Address Lookup Algorithms. IEEE Network Magazine **15:2** (2001) 8-23
7. Sikka, S., Varghese, G.: Memory Efficient State Lookups With Fast Updates. In: Applications, Technologies, Architectures, and Protocols for Computer Communication. ACM SIGCOMM (2000) 335-347