Tight Bounds for Online Class-Constrained Packing [1]

Hadas Shachnai [a,2,3] and Tami Tamir [b]

[a] *Computer Science Department, The Technion, Haifa 32000, Israel.*

[b] *Department of Computer Science and Engineering, Box 352350, Univ. of Washington, Seattle, WA 98195.*

Abstract

We consider *class constrained* packing problems, in which we are given a set of bins, each having a capacity v and c compartments, and n items of M different classes and the same (unit) size. We need to fill the bins with items, subject to capacity constraints, such that items of different classes are placed in separate compartments; thus, each bin can contain items of at most c distinct classes. We consider two optimization goals. In the *class-constrained bin-packing problem (CCBP)*, our goal is to pack all the items in a minimal number of bins; in the *class-constrained multiple knapsack problem (CCMK)*, we wish to maximize the total number of items packed in m bins, for $m > 1$. The CCBP and CCMK problems model fundamental resource allocation problems in computer and manufacturing systems. Both are known to be strongly NP-hard.

In this paper we derive tight bounds for the online variants of these problems. We first present a lower bound of $(1 + \alpha)$ on the competitive ratio of *any* deterministic algorithm for the online CCBP, where $\alpha \in (0, 1]$ depends on v, c, M and n. We show that this ratio is achieved by the algorithm *first-fit*. We then consider the *temporary CCBP*, in which items may be packed for a bounded time interval (that is *unknown* in advance). We obtain a lower bound of v/c on the competitive ratio of *any* (deterministic or randomized) algorithm. We show that this ratio is achieved by all *any-fit* algorithms. Finally, tight bounds are derived for the online CCMK and the *temporary* CCMK problems.

Key words: Class-constraints, Bin Packing, Multiple Knapsack, Temporary Assignment, Online algorithms

1 Introduction

In the well-known *bin packing (BP)* and *multiple knapsack (MK)* problems, a set of items of different sizes and values needs to be packed into bins of limited capacities; a packing is feasible if the total size of the items placed in a bin does not exceed its capacity. We consider the *class-constrained* variants of these problems, which model fundamental resource allocation problems in computer and manufacturing systems. Suppose that all items have the same (unit) size, and the same value; however, the items may be of different *classes (colors)*. Each bin has a capacity and a limited number of compartments. Items of different colors cannot be placed in the same compartment. Thus, the number of compartments in each bin sets a bound on the number of *distinct* colors of items it can accommodate. A packing is feasible if it satisfies the traditional capacity constraint, as well as the *class constraint*.

Formally, the input to our packing problems is a set of items, I, of size $|I| = n$. Each item $a \in I$ has a unit size and a color. Thus, $I = I_1 \cup I_2 \cdots \cup I_M$, where all items in I_i are of color i, $1 \leq i \leq M$. The items need to be placed in identical bins, each having capacity v and c compartments. The output of our packing problems is a *placement*, which specifies the subset of items from each class to be placed in bin j, for any $j \geq 1$. In any feasible placement, at most v items of at most c distinct colors are placed in bin j, for all $j \geq 1$.

We consider the following optimization problems:

The class-constrained bin-packing problem (CCBP), in which our goal is to find a feasible placement of all the items in a minimal number of bins.

The class-constrained multiple knapsack problem (CCMK), in which there are m bins (to which we refer as knapsacks). Our goal is to find a feasible placement, which maximizes the total number of packed items.

The CCMK problem is known to be NP-hard for $c = 2$, and strongly NP-hard for $c \geq 3$ [13]. These hardness results carry over to CCBP. (Clearly, an optimal solution for BP uses m bins *iff* an optimal solution for the knapsack problem with m bins packs all the items.)

In this paper we study the *online* versions of these problems, in which the items arrive as a sequence, one at a time. In each step we get a unit size item

Email addresses: hadas@cs.technion.ac.il (Hadas Shachnai), tami@cs.washington.edu (Tami Tamir).

[1] A preliminary version of this paper appears in the proceedings of the *Fifth Latin American Theoretical Informatics Symposium (LATIN)*, April 2002.

[2] Corresponding author. Author supported in part by Technion V.P.R. Fund, and by the Fund for the Promotion of Research at the Technion.

[3] Part of this work was done while the author was at Bell Laboratories, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974.

of color i, $1 \leq i \leq M$. We need to pack this item before we know any of the subsequent items. Formally, I is given as a sequence $\sigma = a_1, a_2, \ldots$ of length n, such that $\forall k, a_k \in \{1, \ldots, M\}$. The algorithm can base its decision regarding the packing of a_k solely on the knowledge of $a_1, \ldots a_{k-1}$. The decisions of the algorithm are irrevocable, that is, packed items cannot be repacked at later times, and rejected items (in the knapsack problem) cannot be packed later.

Note that, since all items have unit size, the non class-constrained versions of our problems can be solved optimally by a greedy algorithm that packs each arriving item in the bin that was opened last. For the BP problem, this algorithm uses $\lceil n/v \rceil$ bins, which is clearly optimal. For the MK problem, this algorithm packs $\min(n, Mv)$ items, which is also optimal. Since we are interested in instances in which the value of c imposes a restriction on the packing, we assume throughout this paper that $c < \min(M, v)$.

In our study of CCBP and CCMK, we first use the traditional assumption that packed items remain permanently in the system. In the more general case, some items may be temporary. Thus, the input sequence may consist of (i) arrivals of items: each item colored with $i \in \{1, \ldots, M\}$; (ii) departures of items that were packed earlier. Once all the items of the same color leave a bin, the empty compartment can be allocated to arriving items of (possibly) different color. Generally, a departure is associated with specific *item* (of specific color, placed in specific bin). We also consider the special case where departure is associated with a *color*, and we may choose the item of that color to be removed. The resulting *temporary packing problems* are denoted by CCBP^t and CCMK^t. In the CCBP^t problem, our objective is to minimize the overall number of bins used throughout the execution of the algorithm. In CCMK^t our goal is to maximize the total number of items packed by the algorithm.

1.1 Applications

The CCMK and CCBP have several important applications, including storage management for continuous-media data (see, e.g.,[25,12]), scheduling jobs on parallel machines [23] and production planning [9].[4] These applications fall into a large class of resource allocation problems of the following form. Suppose that we have a set of devices, each possessing some amount of a *shared* resource. A user request for the resource can be satisfied by specific device if (i) there is sufficient amount of the resource available on that device; (ii) the device is in configuration to service this type of user. We represent a device as a bin, with the amount of shared resource given by its capacity; the number of compartments in each bin is the number of distinct configurations of the corresponding device. When our goal is to maximize the number of satisfied requests, we get an instance of CCMK. When we wish to minimize the number

[4] A detailed survey is given in [21].

of (identical) devices needed for satisfying all the requests, we get an instance of CCBP.

The *temporary* CCBP and CCMK problems reflect the nature of user requests to hold the resource for a limited amount of time (which is often unknown in advance). In Section 3.2 we discuss a model in which a departure is associated with *a color*, and the algorithm may choose the item in this color to be omitted. This model captures the flexibility of many systems, in providing service from *any* idle device which is in appropriate configuration to handle users of a given type. Consider, for example, the transmission of video programs which reside on a centralized server, to a set of clients (on demand). Upon request for a video, any non-overloaded disk that holds the video file can provide the service. In particular, suppose that some video, f_1, is played from the disks d_1, d_2, to service the requests of the users u_1, u_2; the disk d_2 is overloaded while d_1 is underloaded. When the system completes the transmission from d_1, it may continue servicing u_2 from d_1, thus reducing the load on d_2. In terms of online class-constrained bin-packing, we may choose the bin from which we omit an item, as long as we take an item of the specified color (i.e., a copy of f_1).

1.2 Performance Measures

Let A be an algorithm for CCBP. We denote by $N_A(\sigma)$, $N_{opt}(\sigma)$, the number of bins used by A and an optimal (offline) algorithm, respectively, for packing the items in σ. By standard definition (see, e.g., [6]), we say that A is ρ-competitive, for some $\rho \geq 1$, if for any input sequence, σ, $r_A(\sigma) = N_A(\sigma)/N_{opt}(\sigma) \leq \rho$. Another measure of interest is the *asymptotic worst-case ratio*, given by

$$r_A = \lim \sup_{N_0 \to \infty} (\max_{\sigma} \frac{N_A(\sigma)}{N_0} \mid N_{opt}(\sigma) = N_0\}).$$

In some cases, we study the performance of a packing algorithm, A, on *a set of inputs*. Formally, for a set S, $r_A(S)$ denotes the competitive ratio of A on inputs from S, that is, $\forall \sigma \in S$, $N_A(\sigma) \leq r_A(S) N_{opt}(\sigma)$. In particular, we are interested in $r_A(S_{c,v}(k,h))$, where for given c, v, the set $S_{c,v}(k,h)$ consists of all the input sequences, σ, in which the number of colors, M_σ, satisfies $kc < M_\sigma \leq (k+1)c$ and $hv < |\sigma| \leq (h+1)v$. Our refined analysis for sets of inputs yields tighter performance bounds, that depend on $v, c, |\sigma|$ and M_σ.

In the CCMK problem, we evaluate an algorithm by the number of packed items. Formally, for a placement algorithm A and an input sequence σ, let $n_A(\sigma, m)$ denote the number of items in σ packed by A in the m bins. Let $n_{opt}(\sigma, m)$ denote the maximal number of items in σ that can be packed in the

bins. The *competitive ratio* of A on σ is given by

$$r_A(\sigma) = \inf_{\sigma,m} \frac{n_A(\sigma,m)}{n_{opt}(\sigma,m)}.$$

1.3 Our Results

In this paper, we study the online (temporary) CCBP and CCMK problems. We obtain the best possible results for items of unit sizes and values, and arbitrary number of classes. Our first main contribution (in Section 2) is a tight lower and upper bound of 2 on the competitive ratio of *any* deterministic algorithm for the online CCBP. Tighter bounds are derived for the subsets of instances $S_{c,v}(k,h)$. Specifically, we show that for any deterministic algorithm A_d, $1 < c < v$, $k \geq 0$ and $h \geq k - 1 + \max\{\lceil k/(c-1) \rceil, \lceil (kc+1)/v \rceil\}$,

$$r_d(S_{c,v}(k,h)) \geq 1 + \frac{k+1-\lceil \frac{kc+1}{v} \rceil}{h+1}. \tag{1}$$

Our bound implies that an algorithm may be close to the optimal offline on long sequences that contain relatively small number of colors (e.g., $k = 2$ and $h \gg 1$); however, it may have a ratio of 2 when, on the average, any subsequence of length v contains items of c distinct colors (Take $h = k \gg 1$). We show that a variant of the *first-fit* algorithm, adapted to class-constrained packing, achieves the bound in (1). A greedy algorithm based on partitioning the items into *color-sets* is shown to be efficient as well.

Next, we examine the performance of the well-known set of *any-fit* algorithms in solving our problems. Recall that for classical BP, all *any-fit* algorithms and some other greedy algorithms (such as *next-fit*) have constant competitive ratios [16,17]. We show that this is no longer true in the presence of class-constraints; that is, *next-fit* has the competitive ratio v/c, while some any-fit algorithms have the competitive ratio $\min(v/c, c-1)$.

Our second main contribution (in Section 3) is a tight lower bound of v/c on the competitive ratio of any algorithm for the CCBP^t problem. This bound is valid for both *deterministic and randomized* algorithms. We show that all any-fit algorithms achieve this bound. It may seem that online algorithms for CCBP^t perform better when departures are associated with a *color*, rather than *specific item*. Indeed, in this case, each departure allows some re-packing of the items. However, we show a lower bound of $\min(v/c, c-1)$ on the competitive ratio of all any-fit algorithms. Thus, when $v < c^2 - c$, we get the ratio v/c obtained for departures of specific items. Also, the color-sets algorithm, that achieves the best possible ratio (of 2) when no departures are allowed, is shown to achieve the ratio of v.

A natural question that we address here is whether a-priori knowledge of the *number of items* to be packed, or the number of *distinct colors* in the input sequence, can help. We answer this question in the negative and show that we gain no advantage from a-priori knowledge of these parameters. Specifically, our lower bounds hold for *any* deterministic algorithm, even one that knows n and M_σ in advance, and the algorithms whose competitive ratios are shown to match the lower bounds, do not use the values of n, M_σ. This holds for all the problems considered in this paper.

Finally, in Section 4 we present the results for CCMK and CCMK^t. For CCMK we show an upper bound of c/v on the competitive ratio of any deterministic algorithm. Any greedy algorithm which never rejects an item that can be packed achieves this bound, regardless of the way it packs the items.[5] We also show that there is no competitive algorithm for CCMK^t.

1.4 Related Work

Online bin-packing has been studied since the early 1970's, starting with the classical works of Garey et al. [11] and Johnson [15]. The performance of *first-fit* (FF) and *best-fit* (BF) was analyzed in [17], where it was shown that $r_{FF}, r_{BF} \leq 1.7$. The first lower bound for any online bin packing algorithm was given by Yao in [26]. He showed that $r_A \geq 1.5$, for any algorithm A. The current best lower bound, 1.540, is due to van Vliet [24]. Detailed surveys of online packing results can be found in [6,10].

There is a wide literature on the *offline* bin packing (BP) and the multiple knapsack (MK) problems (see comprehensive surveys in [14,20]). Shachnai and Tamir introduced in [22] the offline CCMK problem and showed that it is NP-hard. The paper presents an approximation algorithm that achieves a factor of $c/(c+1)$ to the optimal. Golubchik et al. [13] derived a tighter bound of $1 - 1/(1+\sqrt{c})^2$ for this algorithm, with a matching lower bound for *any* algorithm for this problem. The paper [21] considers the offline CCMK and CCBP problems with items of *arbitrary* sizes.

When there is only one item (of arbitrary size) from each color, we get a *cardinality-constrained* packing problem. These problems were studied in [8,18] (in the offline case) and in [5] (in the online case). A recent paper by Krumke et al. [19] studies the online *bin coloring* problem, in which we need to pack unit sized items of different colors into a set of identical bins. However, this problem differs from CCBP, both in the way the items are packed and the objective function (the goal is to minimize the maximal number of distinct colors in any of the bins, given that $c = v$).

[5] Such greedy algorithms, also known as *fair*, were previously considered for other packing problems (see, e.g., in [7,3]).

The problem of online load balancing, in scheduling tasks with *unknown durations* on multiple machines, is dual to bin packing with temporary items. Azar, Broder, and Karlin [4] initiated the study of load balancing with temporary tasks. [6] A technique developed in [2] enables to derive a lower bound on the competitive ratio of randomized algorithms, based on existing lower bound for deterministic algorithms. We use the technique to obtain a general lower bound on the competitive ratio of *any* algorithm for CCBP^t (see in Section 3.1.1).

1.5 Notation and Simple Facts

Given a (possibly partial) packing, we say that a bin is *full* if it contains v items; otherwise it is *non-full*. A bin is *occupied* if it contains items of c distinct colors; otherwise it is *non-occupied*. Denote by $C(B)$ the set of colors contained in the bin B. Initially, $\forall B, C(B) = \emptyset$. During the placement of the items, whenever B allocates a compartment to color i, i is added to $C(B)$.

Let $\sigma = a_1, a_2, \ldots$ be a sequence of items to be packed. Clearly, upon arrival, an item a_k of color i needs to be placed in some bin B such that: (i) B is non-full, and (ii) $i \in C(B)$ or B is non-occupied. A bin satisfying these two requirement is *possible* for a_k. An online algorithm needs to determine in which possible bin a_k will be placed. In CCMK, an item can also be rejected. We first give a simple lower bound on $N_{opt}(\sigma)$ for CCBP.

Property 1 *For any $v > c > 1$ and sequence σ, $N_{opt}(\sigma) \geq \max\{\lceil \frac{n}{v} \rceil, \lceil \frac{M_\sigma}{c} \rceil\}$.*

Proof: The total size of the items in σ is $|\sigma| = n$. Thus, even if all the bins are full, the capacity bound for each bin implies that at least $\lceil n/v \rceil$ bins are used. Similarly, the total number of compartments needed for all the items of σ is at least M_σ. Thus, even if all the bins are occupied, at least $\lceil M_\sigma/c \rceil$ bins are used. ■

Note that when $c = 1$ we pack items of different colors in separate bins, and the greedy algorithm is optimal for CCBP; thus, in our discussion of CCBP we assume that $c > 1$. The offline CCMK with identical bins was considered in [22]. The paper shows that when the input sequence σ satisfies $cN \geq M_\sigma + N - 1$, we can fully utilize the N bins. In other words, for packing $n = Nv$ items, we need $N \geq \frac{M_\sigma - 1}{c-1}$ bins. An upper bound on $N_{opt}(\sigma)$ follows.

Property 2 *For any $v > c > 1, \sigma$, $N_{opt}(\sigma) \leq \max\{\lceil \frac{n}{v} \rceil, \lceil \frac{M_\sigma - 1}{c-1} \rceil\}$.*

An algorithm is called *any-fit* if it never opens a new bin for an item that can be packed in one of the open bins. Any-fit algorithms may differ in the way they choose the bin in which the item will be placed.

[6] Comprehensive surveys of previous work on temporary task scheduling are given e.g. in [10], [6] and [1].

The *first-fit* algorithm, denoted by A_F, always packs an arriving item into the first (lowest indexed) possible bin. That is, we place a new item, $a \in I_i$, in the first non-full bin, B, such that either B is non-occupied or $i \in C(B)$. If no such bin exists, we open for a a new bin.

2 Deterministic Algorithms for CCBP

In this section, we present a general lower bound on the competitive ratio of *any* deterministic algorithm and show that *first-fit* achieves this bound. Another greedy algorithm, based on partitioning the items into color-sets, is also shown to be efficient.

The algorithms that we analyze do not use the values of n, M_σ, while our lower bound holds for any deterministic algorithm, even one that knows n and M_σ in advance. We conclude that a-priori knowledge of these parameters cannot improve the performance of deterministic algorithms for CCBP.

2.1 A Lower Bound for any Deterministic Algorithm

Recall that for given $c, v > 1$, an input sequence σ is in $S_{c,v}(k,h)$ *iff* $kc < M_\sigma \leq (k+1)c$ and $hv < |\sigma| \leq (h+1)v$. In other words, $\lceil M_\sigma/c \rceil = k+1$ and $\lceil |\sigma|/v \rceil = h+1$.

Theorem 2.1 *For any deterministic algorithm, A_d, and for any $v > c > 1$, $k \geq 0$, and $h \geq k - 1 + \max\{\lceil \frac{k}{c-1} \rceil, \lceil \frac{kc+1}{v} \rceil\}$,*

$$r_d(S_{c,v}(k,h)) \geq 1 + \frac{k+1-\lceil \frac{kc+1}{v} \rceil}{h+1}.$$

Proof: We show that there exists an input sequence, $\sigma \in S_{c,v}(k,h)$, such that A_d uses at least $k+h+2-\lceil \frac{kc+1}{v} \rceil$ bins to pack σ, while $N_{opt}(\sigma) = h+1$. The sequence σ is constructed online by the adversary according to the way A_d packs the items. We denote by $B_1, B_2, \ldots$ the sequence of bins used by A_d.

Let $x = h - k + 1 - \lceil \frac{kc+1}{v} \rceil \geq 0$. The length of σ is $n = (k+x)v + kc + 1$, thus $\lceil \frac{n}{v} \rceil = k + x + \lceil \frac{kc+1}{v} \rceil = h+1$. The number of distinct colors in σ is $kc+1$. Therefore, $\sigma \in S_{c,v}(k,h)$. We assume that this is known to the algorithm in advance; thus, A_d can use the values h, k in making its deterministic decisions. The sequence σ is constructed as follows.

Step 1. For $i = 1$ to kc:
pack items of color i, until for the first time, an item of color i is placed in one of the bins $B_{k+x+1}, B_{k+x+2}, \ldots$.

Step 2. Pack items of color $kc+1$, until $|\sigma| = (k+x)v + kc + 1$.

Claim 1 *A_d uses at least $h + k + 2 - \lceil \frac{kc+1}{v} \rceil = 2k + x + 1$ bins to pack σ.*

Proof: Clearly, if B_{2k+x+1} is used in Step 1 then at least $2k+x+1$ bins are used for the whole sequence; otherwise, the claim is proved by the following facts (see Figure 1).

- After Step 1, each of the bins $B_{k+x+1}, B_{k+x+2}, \ldots, B_{2k+x}$ contains exactly c items of different colors. This holds since the adversary moves to the next color whenever an item is placed in one of these bins, for each of the kc colors $1, 2, \ldots, kc$.
- The items packed in Step 2 cannot be placed in $B_{k+x+1}, B_{k+x+2}, \ldots, B_{2k+x}$ (which are occupied by items of colors different from $kc+1$); therefore, they must be placed in the bins $B_1, B_2, \ldots, B_{k+x}$ or $B_{2k+x+1}, B_{2k+x+2}, \ldots$.
- Since $|\sigma| = (k + x)v + kc + 1$ and exactly kc items are placed in the bins $B_{k+x+1}, B_{k+x+2}, \ldots, B_{2k+x}$, the remaining $(k + x)v + 1$ items must be placed in $B_1, B_2, \ldots, B_{k+x}$ or $B_{2k+x+1}, B_{2k+x+2}, \ldots$. Even if each of the bins $B_1, \ldots, B_{k+x}$ is full, at least one item must be placed in B_{2k+x+1}.

■

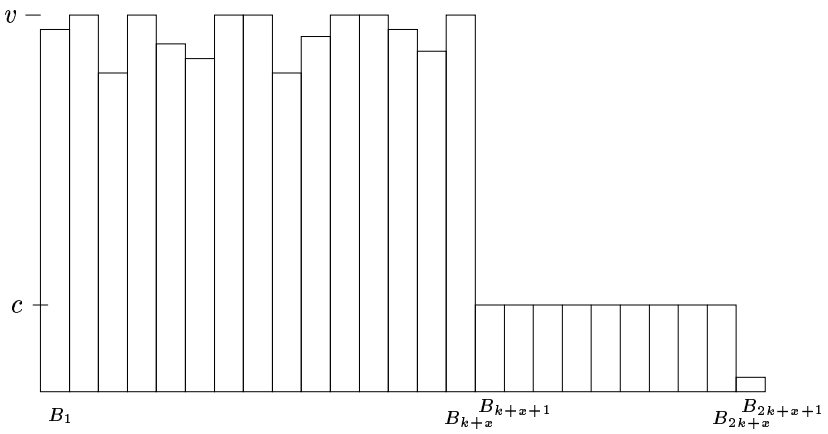

Fig. 1. The placement produced by A_d

Claim 2 *For any $h \geq k - 1 + \lceil \frac{k}{c-1} \rceil$, $N_{opt}(\sigma) = h + 1$.*

Proof: The length of σ is $n = (k + x)v + kc + 1$. The number of distinct colors in σ is $M_\sigma = kc + 1$. By Property 2, $N_{opt}(\sigma) \leq \max\{\lceil \frac{n}{v} \rceil, \lceil \frac{M_\sigma - 1}{c-1} \rceil\}$. We have $\lceil \frac{n}{v} \rceil = h + 1$, and $\lceil \frac{M_\sigma - 1}{c-1} \rceil = \lceil \frac{kc}{c-1} \rceil$. Given that $h \geq k - 1 + \lceil \frac{k}{c-1} \rceil$, we get $h + 1 \geq \lceil \frac{kc}{c-1} \rceil$; that is, $N_{opt}(\sigma) \leq h + 1$. ■

Combining Claims 1 and 2 we conclude that for any c, v, k and $h \geq k - 1 + \max\{\lceil \frac{k}{c-1} \rceil, \lceil \frac{kc+1}{v} \rceil\}$, $r_d(S_{c,v}(k, h)) \geq 1 + \frac{k+1-\lceil \frac{kc+1}{v} \rceil}{h+1}$. ■

Recall that $r_d = \limsup_{N_0 \to \infty}(\max_\sigma\{N_d(\sigma)/N_0 \mid N_{opt}(\sigma) = N_0\})$. For any $N_0 > 1$, let $k = h = N_0 - 1$, and consider an instance with bins having $c \geq k + 1$ and $v \geq kc + 1$. Let σ be the input sequence constructed by the adversary for $S_{c,v}(k, h)$. Note that $h \geq k - 1 + \max\{\lceil \frac{k}{c-1} \rceil, \lceil \frac{kc+1}{v} \rceil\}$. By Claims 1 and 2, A_d uses at least $k + h + 2 - \lceil \frac{kc+1}{v} \rceil = 2N_0 - 1$ bins to pack σ, while

$N_{opt}(\sigma) = h + 1 = N_0$. That is, $r_d(\sigma) \geq \frac{2N_0 - 1}{N_0}$.

Corollary 2.2 *For any deterministic algorithm A_d, $r_d \geq 2$.*

2.2 First-fit, an Optimal Deterministic Algorithm

Consider the *first-fit* algorithm, A_F. Clearly, A_F satisfies the following.

Property 3 *At any time during the execution of the algorithm, each bin, except maybe for the last one, is either full or occupied.*

Note that, given a placement of the items by A_F, for each bin, B, and color, $i \in C(B)$, B allocated a compartment to i only if all the previous bins that contain items of I_i are full. Also, while B is non-full, any $a \in I_i$ can be placed in B. Hence, we get the next result.

Property 4 *In any placement produced by A_F, each non-full bin, B, holds the last item of each of the colors in $C(B)$.*

We show that the competitive ratio of A_F for any input sequence, σ, and for any values of v, c is less than 2. Moreover, for any c, v, k, h, the competitive-ratio of A_F on $S_{c,v}(k,h)$ matches the lower bound given in Theorem 2.1.

Theorem 2.3 *For any c, v, k, h,*

$$r_F(S_{c,v}(k,h)) \leq 1 + \frac{k + 1 - \lceil \frac{kc+1}{v} \rceil}{h+1}.$$

Proof: Let $\sigma \in S_{c,v}(k,h)$ be an input sequence for A_F. Assume that there are m_1 full bins and m_2 non-full bins in the placement produced by A_F for σ. By Property 1, $N_{opt}(\sigma) \geq \lceil \frac{n}{v} \rceil = h + 1$. Thus, $r_F(\sigma) \leq \frac{m_1 + m_2}{h+1}$.

By Property 3, all the non-full bins, except maybe for the last one, are occupied. Therefore, the length of σ is $n \geq m_1 v + (m_2 - 1)c + 1$ (The last non-full bin contains at least one item). Since $\sigma \in S_{c,v}(k,h)$, $h + 1 = \lceil \frac{n}{v} \rceil \geq m_1 + \lceil \frac{(m_2 - 1)c + 1}{v} \rceil$. The coefficients of m_1 and m_2 in this inequality are 1 and $c/v < 1$; thus, $m_1 + m_2$ is maximized when m_2 gets a maximal value.

Claim 3 $m_2 \leq k + 1$.

Proof: By Properties 3 and 4 each non-full bin, except maybe for the last one, contains items of c distinct colors. The last non-full bin contains items of at least one additional color. Thus, $M_\sigma \geq (m_2 - 1)c + 1$. In addition, $M_\sigma \leq (k+1)c$. Thus, $m_2 c \leq (k+2)c - 1$ and since m_2, c are integers $m_2 \leq k+1$. ∎

Setting $m_2 = k + 1$, we get that $m_1 \leq h + 1 - \lceil \frac{kc+1}{v} \rceil$. Thus, $m_1 + m_2 \leq k + 1 + h + 1 - \lceil \frac{kc+1}{v} \rceil$, and $r_F(\sigma) = 1 + \frac{k+1-\lceil \frac{kc+1}{v} \rceil}{h+1}$. ∎

Next we show that for general input sequences, the competitive ratio of A_F is at most 2; thus, by Corollary 2.2, A_F achieves the best possible ratio in the set of deterministic algorithms.

Theorem 2.4 *Let σ be a sequence satisfying $kc < M_\sigma \leq (k+1)c$, for some $k \geq 0$, then $r_F(\sigma) \leq 2 - \frac{1}{k+1}\lceil\frac{kc+1}{v}\rceil$.*

Proof: Given the sequence σ, recall that $h = \lceil|\sigma|/v\rceil - 1$. We distinguish between two cases. If $h \geq k$ then, by Theorem 2.3,

$$r_F(\sigma) \leq 1 + \frac{k+1-\lceil\frac{kc+1}{v}\rceil}{h+1} \leq 1 + \frac{k+1-\lceil\frac{kc+1}{v}\rceil}{k+1} = 2 - \frac{1}{k+1}\lceil\frac{kc+1}{v}\rceil.$$

If $h < k$ then, by Property 1, $N_{opt}(\sigma) \geq \max\{h+1, k+1\} = k+1$. Assume that there are m_1 full bins and m_2 non-full bins in the placement produced by A_F for σ. Thus, $r_F(\sigma) \leq \frac{m_1+m_2}{k+1}$. As in the proof of Theorem 2.3, we have that m_1+m_2 gets its maximal value when $m_2 = k+1$, and $m_1 \leq h+1-\lceil\frac{kc+1}{v}\rceil < k+1-\lceil\frac{kc+1}{v}\rceil$. Therefore, $m_1+m_2 < 2(k+1)-\lceil\frac{kc+1}{v}\rceil$. ∎

2.3 Other Deterministic Algorithms

2.3.1 The Color-Sets Algorithm

Consider a simple algorithm, A_{CS}, which partitions the M_σ colors in σ into $\lceil\frac{M_\sigma}{c}\rceil$ color-sets and packs the items of each color-set greedily. Each color-set consists of c colors (excluding the last color-set, that may contain fewer colors).

The partition into color-sets is determined online by the input sequence. That is, the first set, C_1, consists of the first c colors in σ, the second set, C_2, of the next c colors in σ and so on. At any time, there is one active bin for each color set. When an item a of color $i \in C_j$ arrives, it is placed in the active bin of C_j. If the active bin contains v items, we open a new bin for a and this is the new active bin of C_j. Since $|C_j| \leq c$, the resulting placement is feasible.

Theorem 2.5 *For A_{CS}, the color-sets algorithm, $r_{CS} < 2$.*

Proof: Assume that when A_{CS} terminates there are ℓ active bins, containing $x_1, \ldots, x_\ell$ items. Since we open a new active bin for some color-set only when the current active bin of that color set is full, we have

$$N_{CS}(\sigma) = \frac{n-(x_1+x_2+,\ldots,x_\ell)}{v} + \ell \leq \frac{n}{v} + \ell(1-\frac{1}{v}). \qquad (2)$$

Note that (2) is maximized when ℓ is maximized. Since $\ell \leq \lceil\frac{M_\sigma}{c}\rceil$, we have

$$N_{CS}(\sigma) \leq \frac{n}{v} + \lceil\frac{M_\sigma}{c}\rceil(1-\frac{1}{v}) \leq 2N_{opt}(\sigma) - \frac{1}{v}\lceil\frac{M_\sigma}{c}\rceil.$$

Thus, $r_{CS} < 2$. ∎

2.3.2 Any-fit Algorithms

Recall that an *any-fit* (AF) algorithm never opens a new bin for an item which can be packed in one of the open bins. AF algorithms differ in the way they choose the bin in which the item will be placed. In class-constrained packing, AF algorithms may differ also by the way they select the compartment in which an item of color i will be accommodated. We consider below two types of AF algorithms.

- **Strong any-fit** algorithms, which never allocate a new *compartment* to an item of color i, if a bin that is possible for the item has a compartment of color i.
- **Weak any-fit** algorithms, which may allocate a new *compartment* for an item of color i, even if one of the possible bins has a compartment of that color.

It turns out that almost any AF algorithms falls in the second category, as shown in our next result.

Theorem 2.6 *First-fit is the only strong-AF algorithm.*

Proof: First note that A_F is an any-fit algorithm. We open a new bin for an item only if none of the open bins is possible for that item. Assume that A_F is not a strong-AF algorithm. That is, for some $a \in I_i$, the bin B_j, which is the first possible bin for a, does not have a compartment for I_i, while some other bin B_k, with $k > j$, has a compartment allocated to I_i. Let b be the first item of I_i placed in B_k. Clearly, B_j was possible for b at the time it arrived. A contradiction to the way A_F packs items. Thus, A_F is a strong AF algorithm.

Let A_s be a strong AF algorithm. A_s never allocates a new compartment if an arriving item can be placed in an open compartment. Thus, for each color, i, at any time, there is at most one open possible bin. Specifically, either there is a single occupied bin which holds color i, to which we can add the item, or we need to open a compartment for i. By Property 3, this may be possible only in the last open bin. Being the only possible bin, this bin is also the 'lowest indexed bin among the possible ones'. Thus, the execution of A_s is identical to the one of A_F. ■

In the following we derive upper and lower bounds on the competitive ratios of AF algorithms. To this end, we first prove a general upper bound for a set of algorithms that satisfy Property 3.

Theorem 2.7 *Let A be an algorithm for which Property 3 holds. Then, for any $v > c > 1$, $r_A \leq \min(v/c, c+1)$.*

Proof: We first show that, for any sequence σ, $N_A(\sigma) \leq \frac{v}{c} N_{opt}(\sigma) + 1$. By Property 3, when A terminates, each bin, except maybe for the last one, contains at least c items. Thus, $N_A(\sigma) \leq \frac{n}{c} + 1$. By Property 1, $N_{opt}(\sigma) \geq \frac{n}{v}$. It follows that $N_A(\sigma) \leq \frac{v}{c} N_{opt}(\sigma) + 1$, and $r_A \leq v/c$.

For the upper bound of $c+1$, assume that A has n_f full bins and n_o occupied bins, then $N_A(\sigma) \leq n_f + n_o + 1$. W.l.o.g., assume that the occupied bins are $B_1, \ldots, B_{n_o}$, then each of the occupied bins, B_i, contains a color that is not contained in any of the bins $B_1, \ldots, B_{i-1}$. It follows that $M_\sigma \geq n_o$. On the other hand, $|\sigma| \geq v \cdot n_f$. Thus, $N_{opt}(\sigma) \geq \max(n_f, n_o/c)$. This implies that

$$r_A(\sigma) = \frac{N_A(\sigma)}{N_{opt}(\sigma)} \leq \frac{n_f + n_o + 1}{\max(n_f, n_o/c)}.$$

If $n_f > n_o/c$ then $r_A(\sigma) \leq c+1+1/n_f$; otherwise, $r_A(\sigma) \leq c+1+c/n_o$. Thus, $r_A \leq c+1$. ∎

Note that since AF algorithms open a new bin only if none of the open bins are possible, all AF algorithms satisfy Property 3. Thus, we have

Corollary 2.8 *For any AF algorithm, A_W, $r_W \leq \min(v/c, c+1)$.*

We now derive a matching lower bound on the competitive ratio of AF algorithms. This distinguishes class-constrained bin packing from classic BP, for which *all* AF algorithms have a *constant* competitive ratio of at most 2 [16,17].

Theorem 2.9 *There exists an AF algorithm, A_w, for which $r_w \geq \min(v/c, c-1)$.*

Proof: Consider the weak AF algorithm, A_w, which places a new item in the last (highest indexed) bin into which it can fit. Given the values of $v > c > 1$, for any $j = 0, 1, \ldots$, let σ_j be the sequence of c requests $c+j, 1, 2, \ldots, c-1$. Fix an integer $\ell > 0$, and let $x = \ell(c-1)+1$ and $z = \max(\lceil \frac{xc}{v} \rceil, \lceil \frac{x}{c-1} \rceil)$. Then we construct a sequence σ such that $N_w(\sigma) = x$, while $N_{opt}(\sigma) \leq z$.

The sequence σ, of length xc, consists of the sub-sequences $\sigma_0, \sigma_1, \ldots, \sigma_{x-1}$. A_w packs σ as follows (see Figure 2 for $v = 5, c = 3, \ell = 1$): the c items of σ_0 are placed in the first bin; a new bin is opened for the first item of σ_1, which cannot fit into the first bin. All other items of σ_1 can fit into both the first and the second bin, however, A_w places them all in the second bin. Similarly, for each j, the items of σ_j are placed in a new bin. That is, A_w uses x bins to pack σ.

By Property 2, $N_{opt}(\sigma) \leq \max\{\lceil \frac{n}{v} \rceil, \lceil \frac{M_\sigma - 1}{c-1} \rceil\}$. For the sequence σ, we have $n = xc$ and $M_\sigma = x + c - 1$. Note that $\lceil \frac{M_\sigma - 1}{c-1} \rceil = \lceil \frac{\ell(c-1)+c-1}{c-1} \rceil = \ell + 1$, and that $\lceil \frac{x}{c-1} \rceil = \lceil \frac{\ell(c-1)+1}{c-1} \rceil = \ell + 1$. That is,

$$N_{opt}(\sigma) \leq \max\{\lceil \frac{n}{v} \rceil, \ell + 1\} = \max\{\lceil \frac{xc}{v} \rceil, \lceil \frac{x}{c-1} \rceil\} = z.$$

We get that $r_w(\sigma) \geq \frac{x}{z}$. Note that σ can have arbitrary length, by taking the value of ℓ to be arbitrarily large. Thus,

$$r_w = \lim_{N_0 \to \infty} \sup (\max_{\sigma} \{ \frac{N_w(\sigma)}{N_0} \mid N_{opt}(\sigma) = N_0 \})$$

$$\geq \lim_{\ell \to \infty} \frac{\ell(c-1)+1}{\max\left(\lceil \frac{(\ell(c-1)+1)c}{v} \rceil, \lceil \frac{\ell(c-1)+1}{c-1} \rceil \right)}$$

$$\geq \lim_{\ell \to \infty} \frac{\ell(c-1)+1}{\max\left(\frac{(\ell(c-1)+1)c}{v} + 1, \frac{\ell(c-1)+1}{c-1} + 1 \right)} = \min(\frac{v}{c}, c-1).$$

■

Fig. 2. Lower bound for AF algorithms ($v = 5,\ c = 3$)

2.3.3 The Next-fit Algorithm

The *next-fit* algorithm, denoted by A_N, always packs an arriving item into the currently *active* bin. If the item cannot be placed in the active bin, then the currently active bin becomes inactive (and never used again); a new bin is opened and becomes the active bin. Note that A_N is not an AF algorithm, since it may open a new bin for an item, even if the item can be packed in one of the (inactive) bins that were opened earlier. Yet, since A_N never opens a new bin if the *active* bin is possible for an item, it satisfies Property 3. It follows from Theorem 2.7 that $r_N \leq v/c$. We now derive a matching lower bound for next-fit.

Theorem 2.10 $r_N \geq v/c$.

Proof: Given v, c, for any $N_0 > 1$ consider an input sequence σ, of length $n = N_0 v$, consisting of repetitions of the subsequence $1, 2, \ldots, c+1$. That is, $\sigma = 1, 2, \ldots, c+1, 1, 2, \ldots, c+1, \ldots$. Note that A_N must close each active bin after exactly c items. Thus, $N_N(\sigma) = \lceil \frac{n}{c} \rceil \geq \frac{n}{c}$. On the other hand, since

$M_\sigma = c+1$ and $N_0 > 1$, by Property 2, an optimal algorithm packs σ in $\frac{n}{v} = N_0$ bins (see Figure 3). Thus, $N_N(\sigma) \geq \frac{v}{c} N_{opt}(\sigma)$, and $r_N \geq v/c$. ■

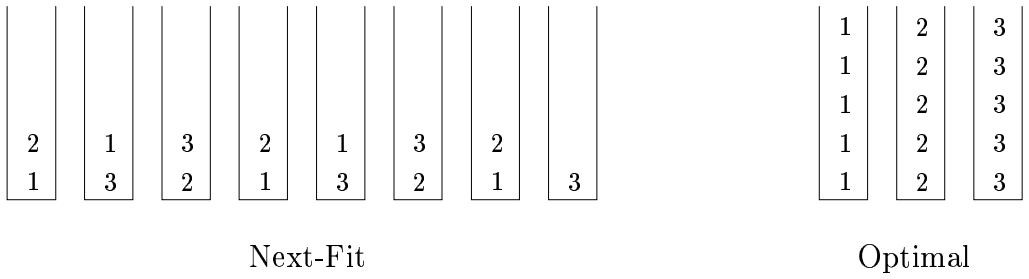

Fig. 3. Lower bound for next-fit ($v = 5, c = 2$, and $N_0 = 3$)

3 Online Class-Constrained Packing of Temporary Items

In this section, we consider a generalization of online class-constrained bin-packing to the case where items may leave the system after a while. We first consider the case where each departure is associated with specific item (of specific color, placed in specific bin). In Section 3.2 we discuss the case in which a departure event is associated with a *color*; thus, we may choose which item of that color to remove. In both cases we do not know in advance the expected departure time of a packed item.

3.1 Departures of Specific Items

In the model where all items are permanent, we showed (in Section 2.2) that *first-fit* is superior to other *any-fit* algorithms, and that its competitive-ratio is less than 2. When some items are temporary, this is no longer true. We prove a lower bound of v/c on the competitive ratio of any deterministic algorithm. Then, we show that all *any-fit* algorithms achieve this bound. Note that, as in Section 2, we conclude that a-priori knowledge of n, M_σ cannot help.

Theorem 3.1 *For any deterministic algorithm A_d, $r_d \geq v/c$.*

Proof: Given v, c, for any N_0 such that $\ell = \frac{1}{c}(N_0 - v + 1)$ is an integer, we construct an input sequence, σ, such that $N_d(\sigma) = \ell v$, while $N_{opt}(\sigma) = \ell c + v - 1 = N_0$. The construction of σ is done in ℓc iterations. In the jth iteration, $1 \leq j \leq \ell c$, we add and remove items of color j. Specifically, the jth iteration in σ consists of:

(1) Arrivals of $v(v-1)+1$ items of color j. These items are placed in at least v distinct bins.
(2) Departures of $v(v-2)+1$ items of color j, selected such that each of the v remaining items of color j is placed in a different bin.

Note that after the iteration ℓc, there are $v\ell c$ items in the bins. Since the items of each color are placed in different bins, each bin holds at most c items. Thus, A_d uses at least ℓv bins.

Consider now an optimal algorithm, A_{opt}, which knows the whole sequence, and in particular, the items that will leave during the second step in each iteration. For each color, A_{opt} will pack the v items that will not be removed in the first available bin, and the $v(v-2)+1$ items that are about to leave, in the next $v-1$ bins. Thus, after the first step of iteration j, $1 \leq j \leq \ell c$, A_{opt} uses $(j-1)+v$ bins, and during the second step of the jth iteration it retreats to j full bins. Hence, the maximal number of bins used by A_{opt} is $\ell c + v - 1$.

We get that $r_d(\sigma) = \frac{\ell v}{\ell c+v-1}$. Recall that ℓ was selected such that $N_0 = \ell c+v-1$. Thus,

$$r_d(\sigma) = \frac{\frac{v}{c}(N_0 - v + 1)}{N_0} = \frac{v}{c}(1 - \frac{v+1}{N_0}),$$

and

$$\lim_{N_0 \to \infty} \sup (\max_{\sigma}\{\frac{N_d(\sigma)}{N_0} \mid N_{opt}(\sigma) = N_0\}) = \frac{v}{c}.$$

■

Theorem 3.2 *If A is an any-fit algorithm, then for any instance with bins having volume v and c compartments, $r_A \leq v/c$.*

Proof: Let σ be an input sequence for A. Let N_A be the number of bins used by A for packing σ. When analyzing the competitive ratio of A, we can assume w.l.o.g. that σ ends when bin number N_A is opened. Indeed, after this point, the number of bins used by an optimal algorithm can only increase, and we need to bound the largest $\frac{N_A(\sigma)}{N_{opt}(\sigma)}$ ratio. Since A is an any-fit algorithm, when N_A is opened, all the other bins are either full or occupied, and therefore each open bin contains at least c items. Let n be the number of packed items at that time. Then $N_A(\sigma) \leq \frac{n}{c}+1$. Clearly, $N_{opt}(\sigma) \geq \frac{n}{v}$. Thus, $N_A(\sigma) \leq \frac{v}{c}N_{opt}(\sigma)+1$, and $r_A \leq v/c$. ■

3.1.1 A Lower Bound for Randomized Algorithms

We now show that randomization cannot help in packing temporary items. Specifically,

Theorem 3.3 *For any randomized algorithm A_R for CCBP^t, $r_R \geq v/c$.*

Proof: Following the technique developed by Azar and Epstein [2], we represent the lower bound for deterministic algorithms (as given in Theorem 3.1) by a set of sequences. Denote this set by T; each sequence is associated with the operation of a deterministic algorithm. These sequences specify the items that leave during the second step of each iteration. The selection of items is adapted to the decisions made by the algorithm. Thus, instead of defining σ online, we include in T all the possible sequences.

Next, we modify the sequence σ, by adding at the end a sequence of departures, of the ℓvc items that remained after the last phase. This extension of σ is applied to all the sequences in T. Hence, for any extended sequence in T, all bins are empty when the sequence ends. Note that since we measure the number of bins used along the execution of σ, rather than the number of open bins at the end, this extension does not affect the number of bins used by A_d or by an optimal algorithm.

Now, we use the adaptation of Yao's theorem for online algorithms. It asserts that a lower bound on the competitive ratio of deterministic algorithms, for any distribution on the input, is also a lower bound for randomized algorithms, given by $E(N_A/N_0)$. We construct randomly a sequence, σ_R, in which on one hand, the value N_0 is the same as the known optimal value for the original non-extended σ, and on the other hand, with high probability, the value of N_A is also the same as the value for the original non-extended σ.

The sequence σ_R is constructed by choosing uniformly at random sequences from T. We call each such sequence *a segment*. We repeat the choice of segments $|T|b$ times, for some $b > 1$, and concatenate the segments to one long sequence. This defines a distribution on the set of possible long sequences. Since the offline optimal algorithm uses the same number of bins for all possible segments ($N_0(\sigma) = \ell c + v - 1$, as in Theorem 3.1), and since any segment ends with all the bins empty, the offline optimal algorithm uses N_0 bins for any σ_R.

Now, for any deterministic algorithm, A, there exists a segment in T for which A uses $N_A(\sigma) = \ell v$ bins. Thus, with probability at least $1/|T|$, the online deterministic algorithm uses $N_A(\sigma)$ bins for a specific segment (and thus for the whole sequence). The probability that less than $N_A(\sigma)$ bins are used is therefore at most $(1 - \frac{1}{|T|})^{|T|b} \leq e^{-b}$. We conclude that with probability at least $1 - e^{-b}$ the competitive ratio of A is $N_A(\sigma)/N_0(\sigma)$, and with probability at most e^{-b} it is at least 1. Let $N_A(\sigma_R)$ and $N_0(\sigma_R)$ be the number of bins used to pack the long sequence σ_R by A and by an optimal algorithm, respectively. Then,

$$E(\frac{N_A(\sigma_R)}{N_0(\sigma_R)}) \geq (1 - e^{-b})\frac{N_A(\sigma)}{N_0(\sigma)} + e^{-b}. \tag{3}$$

For sufficiently large b, the RHS of (3) approaches the ratio $(\ell c + v - 1)/(\ell v)$, and taking $\ell \to \infty$ we get that $r_R \to v/c$. ∎

3.2 Departures of Items of Specific Colors

We now show that any-fit algorithms achieve a poor ratio, even if departures are associated only with a color, and the algorithm may select the item to

be removed in this color. Our result holds for *any* removal policy. Clearly, Theorem 3.2 is valid also for this model, thus, v/c is an upper bound for the competitive-ratio of any-fit algorithms. We show that this bound is tight for bins with $v \leq c(c-1)$.

Theorem 3.4 *For* any *any-fit algorithm* A, $r_A \geq \min(v/c, c-1)$.

Proof: Given v, c, let $x = v - c + 1$, and let $z = \max(\lceil \frac{xc}{v} \rceil, \lceil \frac{x}{c-1} \rceil)$. For any N_0 such that $\ell = \frac{1}{z}(N_0 + z - x)$ is an integer, we construct a sequence σ such that $N_A(\sigma) = \ell x$, while $N_{opt}(\sigma) = \ell z - z + x = N_0$.

The construction of σ is done in ℓ iterations. In the jth iteration ($1 \leq j \leq \ell$) we handle items in the v colors $(j-1)v+1, \ldots, jv$. Specifically, the jth iteration in σ consists of:

(1) Arrivals of xv items that fill the x bins $(j-1)x+1, \ldots, jx$ as follows. For $i = 1$ to x: σ contains a sub-sequence of arrivals of $(c-1)$ items in colors $(j-1)v+1, \ldots, (j-1)v+c-1$, followed by $v-c+1$ arrivals of items of color $(j-1)v+c+i-1$. Since A is an any-fit algorithm, there is at most one possible open bin for each item as it arrives; thus, the bins $(j-1)x+1, \ldots, jx$ are filled sequentially one after the other.

(2) For $i = 1$ to x: remove $v-c$ items of color $(j-1)v+c+i-1$. Now, regardless of the removal policy of the algorithm, we end up with each of the x bins $(j-1)x+1, \ldots, jx$ containing exactly c items, one in each of the colors $(j-1)v+1, \ldots, (j-1)v+c-1$, and one item whose color is in $(j-1)v+c, \ldots, jv$.

Note that after each iteration, $1 \leq j \leq \ell$, each bin contains c items of c different colors. Thus, the arriving items in each iteration, which are of new colors, must be packed in new bins. It follows that after the jth ($1 \leq j \leq \ell$) iteration, A uses jx bins, each containing c items.

Claim 4 *An optimal placement of* σ *uses* $N_0 = \ell z - z + x$ *bins.*

Proof: Consider an optimal algorithm, A_{opt}, which knows the entire sequence, and in particular, the items that will leave during the second step of each iteration. In each iteration, A_{opt} can pack the items that are not removed in the first available bins. We show that these permanent items can be packed in z bins. Note that this set of items consists of xc items of $x+c-1$ colors. We distinguish between two types of colors: (i) *repeated*, from each of which there are x items (In the jth iteration, these are the $(c-1)$ colors $(j-1)v+1, \ldots, (j-1)v+c-1$), and (ii) *single-items*, from each of which there is only one item (in the jth iteration, these are the x colors $(j-1)v+c, \ldots, jv$).

Suppose that $\lceil \frac{xc}{v} \rceil \geq \lceil \frac{x}{c-1} \rceil$, then A_{opt} can pack the permanent items in $\lceil \frac{xc}{v} \rceil$ bins as follows. At first, each bin contains $c-1$ items of single-item colors, and $x = v - c + 1$ items of some repeated color. Once all the 'singles' are packed, we fill the remaining bins greedily with the remaining items of the repeated

colors. Since all the bins are filled to capacity v, A_{opt} uses $\lceil \frac{xc}{v} \rceil$ bins.

Suppose that $\lceil \frac{xc}{v} \rceil < \lceil \frac{x}{c-1} \rceil$, then we can pack the permanent items in $\lceil \frac{x}{c-1} \rceil$ bins as follows. First, we pack in each bin $c-1$ single items, and $x = v-c+1$ items of some repeated color. Once all the items of the repeated colors are packed, we use the remaining bins greedily to pack c single items in each. Since each bin contains at least $c-1$ single items, and there are x such items, all the single items are packed. Also, since $\lceil \frac{xc}{v} \rceil < \lceil \frac{x}{c-1} \rceil$, there is enough capacity for the other items.

Thus, after the first step of iteration j, $1 \leq j \leq \ell$, A_{opt} uses $(j-1)z+x$ bins, and during the second step of the jth iteration it retreats to jz full bins. The maximal number of bins is $(\ell-1)z+x = N_0$, used during the last iteration. ■

We get that $r_A(\sigma) \geq \frac{\ell x}{\ell z - z + x}$, and

$$r_A = \lim_{N_0 \to \infty} \sup (\max_{\sigma} \{ \frac{N_A(\sigma)}{N_0} \mid N_{opt}(\sigma) = N_0 \}) \geq \frac{x}{z} \geq \min(\frac{v}{c}, c-1).$$

■

3.2.1 The Color-Sets Algorithm

We now show that any algorithm based on packing by color sets achieves the ratio v, even if departures are associated only with a color. Our result holds for any removal policy and any algorithm which packs items by color-sets. Note that since each active bin accommodates at least one item, the ratio v is the worst possible.

Theorem 3.5 *Let A_{CS} be a color-sets algorithm with any removal policy, then $r_{CS} \geq v$.*

Proof: Recall that A_{CS} partitions the M_σ colors in σ into $\lceil \frac{M_\sigma}{c} \rceil$ color-sets, where the jth set, C_j, contains the jth set of c colors in σ. The items of each color-set are packed greedily.

Given v and c, for any N_0, let $\ell = N_0 - v + 1$. We construct a sequence σ such that $N_A(\sigma) = \ell v$, while $N_{opt}(\sigma) = \ell + v - 1 = N_0$. W.l.o.g., the colors are numbered by the order of their first appearance in σ; thus, for each $1 \leq j \leq \ell$, $C_j = \{(j-1)c+1, \ldots, jc\}$. The construction of σ is done in ℓ iterations. In the jth iteration, $1 \leq j \leq \ell$, we add and remove items whose colors are in C_j. Specifically, the jth iteration consists of two steps:

(1) Arrivals of v^2 items whose colors are in C_j. This sequence consists of repeating v times a sub-sequence of arrivals of a single item of color jc, followed by $v-1$ arrivals of items whose colors are in the set $\{(j-1)c+1, \ldots, jc-1\}$. The sequence contains at least one item in each of the colors in C_j (so A_{CS} can define C_j).
(2) Departures of the $v(v-1)$ items in the colors $(j-1)c+1, \ldots, jc-1$.

Note that in the first step of iteration j, A_{CS} fills the bins $(j-1)v+1, \ldots, jv$, and after the second step of iteration j, regardless of the removal policy of A_{CS}, each of these bins contains a single item (of color jc). However, these bins cannot be used by A_{CS} in the next iterations, since the arriving items belong to different color-sets. Thus, in each iteration, the arriving items are packed in new bins. It follows that after the ℓth iteration, A_{CS} has ℓv open bins, each contains a single item.

Consider now an optimal algorithm, A_{opt}, which knows the entire sequence, and in particular, the items that will leave in the second step of each iteration. In each iteration, A_{opt} packs the v permanent items in the first available bin and the temporary items in the next $v-1$ bins. Thus, after the first step of iteration j, $1 \leq j \leq \ell$, A_{opt} has $j-1+v$ full bins, and during the second step of the jth iteration it retreats to j full bins. The maximal number of bins is $\ell+v-1$, used during the last iteration.

We get that $r_{CS}(\sigma) = \frac{\ell v}{\ell+v-1}$. Since $N_0 = \ell+v-1$, $r_{CS}(\sigma) = \frac{N_0 v - v^2 + v}{N_0} = v(1 - \frac{v^2-v}{N_0})$, and $\limsup_{N_0 \to \infty}(\max_\sigma\{N_{CS}(\sigma)/N_0 \mid N_{opt}(\sigma) = N_0\}) = v$. ■

Remark: The same ratio of v can be shown also for variants of A_{CS} that use different partition rule (that is, the color sets are not necessarily determined by the *first* appearance of each color in σ). In this case, if the algorithm switches during the jth iteration to a new color-set (i.e., an arriving item is assigned to C_r, $r > j$), then the item in C_r is temporarily 'ignored' by the adversary, and the jth iteration continues as described above. The item in C_r will be handled in the rth iteration.

4 The Online Class-Constrained Multiple Knapsack Problem

Recall that in the CCMK problem we have m identical knapsacks, of volume v and c compartments, and our goal is to maximize the number of packed items from the input sequence σ.

In this section, we show that the best possible competitive ratio for a deterministic algorithm for CCMK is c/v. This bound is achieved by any greedy algorithm, i.e., an algorithm that rejects an arriving item only if it cannot be packed in any of the knapsacks. We show that any greedy algorithm achieves the ratio c/v, regardless of the way the items are packed. For the temporary CCMK problem, note that the number of packed items may be larger than mv. We show that no deterministic algorithm is competitive.

Theorem 4.1 *For any $v \geq c$, and any deterministic algorithm, A_d, for CCMK, $r_d \leq c/v$.*

Proof: Consider the following sequence that is constructed online by the adversary for an algorithm A_d.

(1) For all $i = 1, \ldots, mc$, repeat items of color i until one such item is packed, or until mv items of color i are rejected. In the latter case, the sequence ends.
(2) mv items of color $mc + 1$.

If no color is rejected in mv steps, then each of the knapsacks filled by A_d contains exactly c items of c distinct colors, thus, only mc items are packed. An optimal algorithm can pack the mv items of color $mc+1$. If some color, i, is persistently rejected, then an optimal algorithm can pack the mv rejected items of color i, while A_d packs only $i-1$ items of the colors $1, \ldots, i-1$. In both cases $r_d \leq \frac{mc}{mv} = c/v$. ∎

A greedy algorithm never rejects an item that can be packed, at the time it arrives. We show that the competitive ratio of any greedy algorithm matches the c/v bound, regardless of the way the items are packed.

Theorem 4.2 *For any greedy algorithm A_G, the competitive ratio of A_G is $r_G \geq c/v$.*

Proof: For any $m > 0$ and a sequence σ, consider the knapsacks when σ terminates. If some knapsack contains less than c items, then, since A_G is greedy, no item was rejected and $r_G(\sigma, m) = 1$. Otherwise, there are at least c items in each of knapsack, and $n_G(\sigma, m) \geq cm$. Since $n_{opt}(\sigma, m) \leq vm$, we get the desired ratio. ∎

Next we show that when some of the items are temporary no algorithm is competitive.

Theorem 4.3 *For any $v \geq c$, any deterministic algorithm, A_d, for $CCMK^t$, and any $\rho > 0$, the competitive ratio of A_d is $r_d \geq \rho$.*

Proof: Consider a sequence, σ, that first 'blocks' all the m knapsacks (as in the proof of Theorem 4.1). After all the knapsacks are full or occupied, σ continues with a sequence of arbitrary length, ρ, of pairs of arrival-departure of some item (of a new color). Clearly, these repeated arrivals cannot be accepted by the algorithms, while an optimal algorithm can pack them all. Any ratio can be obtained by picking the value of ρ large enough. ∎

Acknowledgment

We thank an anonymous referee for many insightful comments on the paper.

References

[1] A. Armon, Y. Azar, L. Epstein, and O. Regev, Temporary tasks assignment resolved. In *Proc. of the 13th ACM-SIAM Symposium on Discrete Algorithms*

(SODA), pages 116–124, 2002.

[2] Y. Azar and L. Epstein, On-line load balancing of temporary tasks on identical machines. Proc of 5th ISTCS, 119–125, 1997.

[3] Y. Azar, J. Boyar, L. M. Favrholdt, K. S. Larsen, M. N. Nielsen and L. Epstein. Fair versus unrestricted bin packing. Algorithmica 34(2): 181–196, 2002.

[4] Y. Azar, A. Broder and A. Karlin. On-line load balancing, Theoretical Computer Science, vol. 130, pages 73–84, 1994.

[5] L. Babel, B.Chen, H. Kellerer and V. Kotov. On-line algorithms for cardinality constraint bin packing problems. Technical report, Institut fuer Statistik und Operations Research, Universitaet Graz, 2001.

[6] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 1998.

[7] J. Boyar, K. S. Larsen, M. N. Nielsen The accommodating function: a generalization of the competitive ratio. *SIAM J. on Computing*, 31(1): 233–258, 2001.

[8] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operations Research*, 123:333–345, 2000.

[9] W. J. Davis, D. L. Setterdahl, J. G. Macro, V. Izokaitis, and B. Bauman. Recent advances in the modeling, scheduling and control of flexible automation. In *Proc. of the Winter Simulation Conference*, pages 143–155, 1993.

[10] A. Fiat and G.J. Woeginger. *Online Algorithms: The State of the Art.* LNCS. 1442, Springer-Verlag, 1998.

[11] M.R. Garey, R.L.Graham, and J.D.Ullman. Worst-case analysis of memory allocation algorithms. In *Proc. of the 4th ACM Symposium on theory of Computing*, 1972.

[12] S. Ghandeharizadeh and R.R. Muntz. Design and implementation of scalable continuous media servers. *Parallel Computing Journal*, 24(1):91–122, 1998.

[13] L. Golubchik, S. Khanna, S. Khuller, R. Thurimella, and A. Zhu. Approximation algorithms for data placement on parallel disks. In *Proc. of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 223–232, 2000.

[14] D.S. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* PUS Publishing Company, 1995.

[15] D.S. Johnson. *Near-optimal bin packing algorithms.* PhD thesis, MIT, Cambridge, MA, 1973.

[16] D.S. Johnson. Fast algorithms for bin packing. *J. Comput. System Sci.*, 8:272–314, 1974.

[17] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, and R.L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithm. *SIAM Journal of Computing*, 3:256–278, 1974.

[18] H. Kellerer and U. Pferschy. Cardinality constrained bin-packing problems. *Annals of Operations Research*, 92:335–349, 1999.

[19] S.O. Krumke, W. de Paepe, J. Rambau and L. Stougie. Online bin-coloring. In *Proc. of the 9th Annual European Symposium on Algorithms (ESA)*, 2001.

[20] S. Martello and P. Toth. Algorithms for knapsack problems. *Annals of Discrete Math.*, 31:213–258, 1987.

[21] H. Shachnai and T. Tamir. Polynomial time approximation schemes for class-constrained packing problems. *Journal of Scheduling*, 4:313–338, 2001.

[22] H. Shachnai and T. Tamir. On two class-constrained versions of the multiple knapsack problem. *Algorithmica*, 29:442–467, 2001.

[23] H. Shachnai and T. Tamir. Multiprocessor scheduling with machine allotment and parallelism constraints. 2001. *Algorithmica*, Vol. 32:4, 651–678, 2002.

[24] A. van Vliet. On the asymptotic worst case behavior of harmonic fit. *J. of Algorithms*, 20:113–136, 1996.

[25] J.L. Wolf, P.S. Yu, and H. Shachnai. Disk load balancing for video-on-demand systems. *ACM Multimedia Systems Journal*, 5:358–370, 1997.

[26] A. Yao. New algorithms for bin packing. *Journal of the ACM*, 27:207–227, 1980.