Theoretical Computer Science 602 (2015) 103–113

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Online bin stretching with bunch techniques

Michaël Gabay [a,b,*], Vladimir Kotov [c], Nadia Brauner [a,b]

[a] Univ. Grenoble Alpes, G-SCOP, F-38000 Grenoble, France
[b] CNRS, G-SCOP, F-38000 Grenoble, France
[c] Belarusian State University, FPMI DMA Department, 4 Nezavisimosti Avenue, 220030 Minsk, Belarus

ARTICLE INFO

Article history:
Received 4 October 2013
Received in revised form 6 June 2015
Accepted 28 July 2015
Available online 21 August 2015
Communicated by T. Erlebach

Keywords:
Bin stretching
Multiprocessor scheduling
Online algorithms
Bin packing

ABSTRACT

We are given a sequence of items that can be packed into m unit size bins and the goal is to assign these items online to m bins while minimizing the stretching factor. Bins have infinite capacities and the stretching factor is the size of the largest bin. We present an algorithm with stretching factor $26/17 \approx 1.5294$ improving the best known algorithm by Kellerer and Kotov (2013) [1] with a stretching factor $11/7 \approx 1.5714$. Our algorithm has 2 stages and uses bunch techniques: we aggregate bins into batches sharing a common purpose.

1. Introduction

In bin packing problems, a set of items is to be packed into identical bins of size one; the goal is to minimize the number of bins. We are interested in the online variant of this problem: the items arrive consecutively and each of them must be packed irrevocably into a bin, without any knowledge on future items. Recent research has focused on studying scenarios where some information is known in advance.

We consider the online problem where we know in advance that the items can be packed into m bins of size 1. The objective is to pack the items on arrival into m stretched bins, *i.e.* bins of size at most $\beta = 1 + \alpha$ where β is called the stretching factor. Formally speaking, a bin-stretching algorithm is defined to have a stretching factor β if, for every sequence of items that can be assigned to m bins of unit size, the algorithm successfully packs the items into m bins of size at most β. The goal is to find an algorithm with the smallest possible stretching factor.

This problem was introduced by Azar and Regev [2]. They described a practical application of transferring files on a remote system and remarked that this problem is equivalent to the online makespan minimization problem on identical parallel machines with known value of the optimal makespan.

Graham [3,4] gave the first deterministic online algorithm for this online scheduling problem. He showed that the famous List scheduling algorithm is $(2 - 1/m)$-competitive. A long list of improved algorithms has since been published, the best one is due to Fleischer and Wahl [5].

For the semi-online case, the algorithm is provided with some information on the job sequence or has some extra ability to process it such as decreasing order [4,6,7], known total processing time [8–11], or known number of necessary bins [2] as in our case.

* Corresponding author.
E-mail addresses: michael.gabay@g-scop.grenoble-inp.fr (M. Gabay), kotovvm@bsu.by (V. Kotov), Nadia.Brauner@g-scop.grenoble-inp.fr (N. Brauner).

http://dx.doi.org/10.1016/j.tcs.2015.07.065
0304-3975/

Notice that the bin stretching problem is different from the semi-online scheduling problem with known total processing time. A simple proof of this statement is that Albers and Hellwig [11] proved that 1.585 is a lower bound for the semi-online scheduling problem with known total processing time while Kellerer and Kotov [1] developed an algorithm with stretching factor $11/7 \approx 1.5714 < 1.585$ for the online bin stretching problem. Until recently, 4/3 was the best known lower bound for the bin stretching problem. This bound is obtained with 2 bins, on input (1/3, 1/3, 1) or (1/3, 1/3, 2/3, 2/3) and can be generalized to any number of bins [2]. A better lower bound of $19/14 \approx 1.3571$ for 3 and 4 bins is presented in [12].

Generalizations of the bin stretching problem includes bin stretching with different machine speeds. The case with 2 uniform machines was studied in [13,14].

In this paper we present an algorithm that uses bunch techniques and provides a stretching factor $26/17 \approx 1.5294$. Recently, Böhm et al. [15] improved the upper bound to 1.5 for any number of bins and to 1.375 for 3 bins, building on the techniques presented in this paper.

1.1. Problem definition and notation

We are given a set of m identical unit size bins and a sequence of n items. Item j has a weight $w_j > 0$ and each item has to be assigned online to a bin. We define the weight of a bin B, denoted by $w(B)$, as the sum of the weights of all items assigned to B. In the course of the algorithm, we define some structures made up of one or several bins. For a given structure S, we denote by $w(S)$ the sum of the weights of all items packed into the bins composing S and $|S|$ is the number of bins in S.

The number m of bins is given as part of the initial input and it is certified that all items can fit into m unit-sized bins. However, we have no more information in the initial input (e.g. the total number of items n is unknown until the end of the input).

We divide the items into 4 disjoint classes as in Table 1 and Fig. 1. Items with weight in $(0; \frac{9}{34}]$ are called *tiny*, items in $(\frac{9}{34}; \frac{9}{17}]$ are called *small*, items in $(\frac{9}{17}; \frac{13}{17}]$ are called *medium* and items in $(\frac{13}{17}; 1]$ are called *large*.

Table 1
Item classes.

Item class	*Tiny*	*Small*	*Medium*	*Large*
Item weight	(0; 9/34]	(9/34; 9/17]	(9/17; 13/17]	(13/17; 1]

In the sequel, we design an algorithm with stretching factor $\frac{26}{17}$. Hence, each bin has a capacity $\frac{26}{17}$ and we say that an item j *fits* into a bin B (or equivalently that packing item j into bin B is *feasible*) if $w(B) + w_j \leq \frac{26}{17}$.

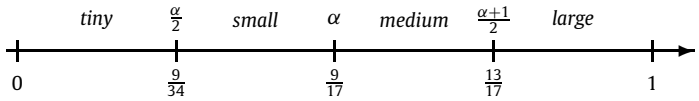

Fig. 1. Item types for a stretching factor of $\beta = 1 + \alpha = \frac{26}{17}$.

1.2. Algorithm overview

We design a two-stage algorithm. In the first stage, we *open* the bins and create *bunches* which we use to fit the items. In the second stage, we fit the items into the remaining *non-reduced* bins and bunches.

In the algorithm, we use different types of bin structures and qualify them as *open*, *closed* or *reduced*. A structure is a group of one or several bins associated with a qualifier. We say that a bin is *open* if it can be used during current stage of the algorithm. A bin is *closed* once it contains enough items. The *closed* status simply means that the function of the bin changes. Closed bins can be reopened and converted into a new structure anytime. Finally, a bin is *reduced* if it will not be used anymore. Any reduced structure S has the property that the sum of the weights of its items is greater than its number of bins: $w(S) \geq |S|$ and for any bin $B \in S$, $w(B) \leq \frac{26}{17}$. Notice that if all bins have been reduced then there is no item remaining and the stretching factor of the current solution is at most $\frac{26}{17}$.

We denote respectively by sB, mB and lB single bins whose first goal is to contain *small*, *medium* and *large* items. $\mathcal{TB}$ and $\mathcal{LB}$ denote bunches intended to contain respectively *tiny* and *large* items. These bins and bunches can also contain different items as we will see later.

A bunch is a group of 4 bins. The aim of these structures is to help fitting items with more flexibility and then reduce them when structure's total weight is greater than or equal to 4. When a new bunch is created, we first assign a single bin to the bunch, then a second one, a third one and eventually the fourth bin. Once 4 bins have been assigned to a bunch, the bunch is complete and its status changes to *closed*. Otherwise, the bunch is incomplete and is denoted by $\mathcal{TB}^i$ where $i \leq 3$ is the number of bins currently assigned to the bunch.