Theoretical Computer Science 556 (2014) 71–84

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Online bin covering: Expectations vs. guarantees ☆

Marie G. Christ, Lene M. Favrholdt, Kim S. Larsen *

Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark

ARTICLE INFO

Article history:
Received 28 February 2014
Received in revised form 12 May 2014
Accepted 18 June 2014
Available online 25 June 2014

Keywords:
Online algorithms
Bin covering
Performance measures
Competitive analysis

ABSTRACT

Bin covering is a dual version of classic bin packing. Thus, the goal is to cover as many bins as possible, where covering a bin means packing items of total size at least one in the bin. For online bin covering, competitive analysis fails to distinguish between most algorithms of interest; all "reasonable" algorithms have a competitive ratio of $\frac{1}{2}$. Thus, in order to get a better understanding of the combinatorial difficulties in solving this problem, we turn to other performance measures, namely relative worst order, random order, and max/max analysis, as well as analyzing input with restricted or uniformly distributed item sizes. In this way, our study also supplements the ongoing systematic studies of the relative strengths of various performance measures.

Two classic algorithms for online bin packing that have natural dual versions are Harmonic$_k$ and Next-Fit. Even though the algorithms are quite different in nature, the dual versions are not separated by competitive analysis. We make the case that when guarantees are needed, even under restricted input sequences, dual Harmonic$_k$ is preferable. In addition, we establish quite robust theoretical results showing that if items come from a uniform distribution or even if just the ordering of items is uniformly random, then dual Next-Fit is the right choice.

1. Introduction

Bin covering [2] is a dual version of classic bin packing. As usual, bins have size one and items with sizes between zero and one must be packed. However, in bin covering, the objective is to cover as many bins as possible, where a bin is covered if the sizes of items placed in the bin sum up to at least one. We are considering the online version of bin covering. A problem is online if the input sequence is presented to the algorithm one item at a time, and the algorithm must make an irrevocable decision regarding the current item without knowledge of future items.

Bin covering algorithms have numerous important applications. For instance, when packing or canning food items guaranteeing a minimum weight or volume, reductions in the overpacking of even a few percent may have a large economic impact. If items arrive on a conveyor belt, for instance, the problem becomes online.

Classic algorithms for online bin packing are Next-Fit and the parameterized family Harmonic$_k$ [21]. Next-Fit is a very simple and natural algorithm, and Harmonic$_k$ was designed to obtain a competitive ratio [19,24] better than any Any-Fit algorithm (First-Fit and Best-Fit are examples of Any-Fit algorithms for bin packing, and the competitive ratio of Next-Fit is worse than both these algorithms). Harmonic$_k$ and variations of it have been analyzed extensively [22,23,25].

☆ A preliminary version of this paper appeared in the proceedings of the Seventh Annual International Conference on Combinatorial Optimization and Applications, 2013. Supported in part by the Danish Council for Independent Research and the Villum Foundation.

* Corresponding author.
E-mail addresses: christm@imada.sdu.dk (M.G. Christ), lenem@imada.sdu.dk (L.M. Favrholdt), kslarsen@imada.sdu.dk (K.S. Larsen).

http://dx.doi.org/10.1016/j.tcs.2014.06.029
0304-3975/

We consider the obvious dual version of these, DNF [2] and DH_k [12]. These algorithms are quite different in nature and the bin packing versions are clearly separated, having competitive ratios of 2 and approximately 1.691, respectively. However, for bin covering, competitive analysis does not distinguish between them! In fact, for bin covering, competitive analysis categorizes both algorithms as being optimal among deterministic algorithms, but also worst possible among "reasonable" algorithms for the problem. This is unlike the situation in bin packing, and in general, results from bin packing do not transfer directly to bin covering.

To understand the algorithmic differences better, it is therefore necessary to employ different techniques, and we turn to other generally applicable performance measures, namely relative worst order analysis, random order analysis, and max/max analysis. As for almost all performance measures, the idea is to abstract away some details of the problem to enable comparisons. Without some abstraction, it is hard to ever, analytically, claim that one algorithm is better than another, since almost any algorithm performs better than any other algorithm on at least one input sequence. For all the measures considered here, the abstraction can be viewed as being defined via first a partitioning of the set of input sequences of a given length and then an aggregation of the results from each partition. For each sequence length, competitive analysis, for instance, considers all the ratios of the online performance to the optimal offline performance obtained for each sequence of that length, and then takes the worst ratio of all of these. The measures above employ a less fine-grained partitioning of the input space. Worst order and random order analysis group permutations of the same sequence together instead of considering each sequence separately, deriving worst-case or average-case performance, respectively, within each partition. With max/max analysis the partitioning of the input space is even coarser: for each sequence length n, the online worst-case behavior over all sequences of length n is compared to the worst-case optimal offline behavior over all sequences of length n. There is no one correct way to compare algorithms, but since these measures focus on different aspects of algorithmic behavior, considering all of the ones above lead to a very broad analysis of the problem. Extensive motivational sections can be found in the papers introducing these measures and in the survey [13]. As a further supplement, we analyze restricted input sequences, where items have similar size, which is likely to happen in practice if one is packing products with an origin in nature, for instance. Finally, we consider input sequences containing items having uniformly distributed sizes.

Relative worst order analysis [4,5] has been applied to many problems; a recent list can be found in [15]. In [16], bin covering was analyzed, but using a version of the problem allowing items of size 1. We analyze the more commonly studied version for bin covering, where all items are strictly smaller than 1. Since worst-case sequences from [16] contain items of size 1, this leads to slightly different results. For completeness, we include these results. Random order analysis [20] was introduced for classic bin packing, but has also been used for other problems; a server problem, for instance [8]. Max/max analysis [3] was introduced as an early step towards refining the results from competitive analysis for paging and a server problem.

Relative worst order analysis emphasizes the fact that there exist multisets of input items where DNF can perform $\frac{3}{2}$ times as poorly as DH_k. On the other hand, DH_k's method of limiting the worst-case also means that it has less of an opportunity to reach the best case, as opposed to DNF. This is reflected in the random order analysis, where DNF comes out at least as well as DH_k. Another way of approaching randomness is to analyze a uniform distribution. We establish new results on DH_k showing that its performance here is slightly worse than that of DNF, in line with the random order results. With the max/max analysis, a distinction between the two algorithms can only be achieved, when the item sizes are limited, and DH_k is the algorithm selected as best by this measure. With respect to competitive analysis, we also consider restricted input in the sense that item sizes may only vary across one or two consecutive DH_k partitioning points. This is a formal way of treating the case where items are of similar size, while allowing greater variation when this size is large. We show that with this restricted form of input, considering the worst-case measure of competitive analysis, DH_k is deemed better than DNF, as DNF is more vulnerable to worst-case sequences.

This study also contributes to the ongoing systematic studies of the relative strengths of various performance measures, initiated in [8]. Up until that paper, most performance measures were introduced for a specific problem to overcome the limitations of competitive analysis. In [8], comparisons of performance measures different from competitive analysis were initiated, and this line of work has been continued in [6,7,9], among others. Our results supplement results in [11], showing that no deterministic algorithm for the bin covering problem can be better than $\frac{1}{2}$-competitive and giving an asymptotically optimal algorithm for the case of items being uniformly distributed on $(0, 1)$. For DNF, [10] established an expected competitive ratio of $\frac{2}{e}$ under the same conditions.

In the following, we formally define the bin covering problem and the algorithms DNF and DH_k, the performance of which we compare under different performance measures. The performance measures themselves are defined in each their section. We conclude on our findings in the final section.

1.1. Bin covering

In the one dimensional bin covering problem, the algorithm gets an input sequence $I = \langle i_1, i_2, \ldots \rangle$ of item sizes, where for all j, $0 < i_j < 1$. The items are to be packed in bins of size 1. A bin is *covered*, if items of total size at least 1 have been packed in it, and the goal is to cover as many bins as possible.

Requiring items to be strictly smaller than 1 corresponds to assuming that items of size 1 are treated separately. This makes sense, since there is no advantage in combining an item of size 1 with any other items in a bin. In other words, any algorithm not giving special treatment to items of size 1 could trivially be improved by doing so.