Theoretical Computer Science 596 (2015) 12–22

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Locality-preserving allocations problems and coloured bin packing

Andrew Twigg [a,1], Eduardo C. Xavier [b,*,2]

[a] Computing Laboratory, University of Oxford, United Kingdom
[b] Institute of Computing, University of Campinas (UNICAMP), Brazil

ARTICLE INFO

Article history:
Received 24 June 2014
Received in revised form 28 April 2015
Accepted 18 June 2015
Available online 25 June 2015
Communicated by T. Erlebach

Keywords:
Bin packing
Approximation algorithms
Locality preserving allocations

ABSTRACT

We study the following problem, introduced by Chung et al. in 2006. We are given, online or offline, a set of coloured items of different sizes, and wish to pack them into bins of equal size so that we use few bins in total (at most α times optimal), and that the items of each colour span few bins (at most β times optimal). We call such allocations (α, β)-approximate. As usual in bin packing problems, we allow additive constants and consider (α, β) as the asymptotic performance ratios. We prove that for $\varepsilon > 0$, if we desire small α, no scheme can beat $(1+\varepsilon, \Omega(1/\varepsilon))$-approximate allocations and similarly as we desire small β, no scheme can beat $(1.69103, 1+\varepsilon)$-approximate allocations. We give offline schemes that come very close to achieving these lower bounds. For the online case, we prove that no scheme can even achieve $(O(1), O(1))$-approximate allocations. However, a small restriction on item sizes permits a simple online scheme that computes $(2+\varepsilon, 1.7)$-approximate allocations.

1. Introduction

We consider the problem of computing locality-preserving allocations of coloured items to bins, so as to preserve locality (colours span few bins) but remain efficient (use a few total bins). The problem appears to be a fundamental problem arising in allocating files in peer-to-peer networks, allocating related jobs to processors, allocating related items in a distributed cache, and so on. The aim is to keep the communication overhead between items of the same colour small. One application for example appears in allocating jobs in a grid computing system. Some of the jobs are related in such a way that results computed by one job are used by another one. There are also non-related jobs that may be from different users and contexts. Related jobs are of a same colour and each job has a length (number of instructions for example). In the grid environment each computer has a number of instructions donated by its owner to be used by the grid jobs. This way the objective is to allocate jobs to machines trying to use few machines (bins) respecting the number of instructions available (bins size), while also trying to keep related jobs together in as few machines as possible. In peer-to-peer systems a similar problem also appears where one wants to split pieces of files across several machines, and wants to keep pieces of a file close together to minimize the time to retrieve the entire file.

* Corresponding author.
E-mail addresses: andy.twigg@comlab.ox.ac.uk (A. Twigg), eduardo@ic.unicamp.br (E.C. Xavier).
[1] Supported by a Junior Research Fellowship, St John's College.
[2] Supported by FAPESP and CNPq.

http://dx.doi.org/10.1016/j.tcs.2015.06.036

These problems can be stated as a fundamental bi-criteria bin packing problem. Let I be a set of items, each item of some colour $c \in C$, and denote by I_c the set of items of a given colour c. Denote by $\mathrm{OPT}(I)$ the minimum number of bins necessary to pack all items and denote by $\mathrm{OPT}(I_c)$ the minimum number of bins necessary to pack only items of colour c, i.e., as if we had a bin packing instance with items I_c. Let $A(I)$ be the number of bins generated by algorithm A when packing all items, and for each colour c, let $A(I_c)$ be the number of bins of this packing having items of colour c. We say that items of colour c span $A(I_c)$ bins in this packing. We want an algorithm that minimizes both ratios $\frac{A(I)}{\mathrm{OPT}(I)}$ and $\max_{c \in C} \frac{A(I_c)}{\mathrm{OPT}(I_c)}$. So we would like to allocate the items to bins so that we use few bins in total (at most $\alpha\mathrm{OPT}(I)$, where we call α the *bin stretch*), and the items of each colour c span few bins (at most $\beta\mathrm{OPT}(I_c)$, where we call β the *colour stretch*). We call such allocations (or packings) (α, β)-approximate. The problem of minimizing any one of α or β is equivalent to the classical one-dimensional bin packing, but as we show, in general it is not even possible to minimize them simultaneously. A natural extension is to consider bins as nodes of some graph G, and we want to allocate bins so that each subgraph G_c induced by nodes containing items of colour c has some natural property allowing small communication overhead, such as having low diameter, or small size.

We prove that for $\varepsilon > 0$, if we desire small bin stretch, no scheme can beat $(1 + \varepsilon, \Omega(1/\varepsilon))$-approximate allocations and similarly as we desire small colour stretch, no scheme can beat $(1.69103, 1 + \varepsilon)$-approximate allocations. We give offline schemes that are based in well know bin packing algorithms and yet come very close to achieving these lower bounds. We show how to construct $(1 + \varepsilon, \Omega(1/\varepsilon))$ and $(1.7, 1 + \varepsilon)$ approximate allocations, the first one closing the gap with the lower bound and the last one almost closing the gap. For the online case, we prove that no scheme can even achieve $(O(1), O(1))$-approximate allocations. However, a small restriction on item sizes permits a simple online scheme that computes $(2 + \varepsilon, 1.7)$-approximate allocations.

2. Preliminaries

We now formulate the problem of computing locality-preserving allocations as a coloured bin packing problem. We are given a set I of n coloured items each item e with a size $s(e)$ in $(0, 1]$ and with a colour $c(e)$ from $C = \{1, \ldots, m\}$, and an infinite number of unit-capacity bins. Let I_c be the set of colour-c items, and denote by $\mathrm{OPT}(I)$ ($\mathrm{OPT}(I_c)$ respectively) the smallest possible number of bins needed to store items in I (I_c respectively). For a packing P of items I, define $P(I)$ as the number of bins used to pack I, and define $P_c(I)$ as the number of bins spanned by colour-c items in the packing P. When I is obvious, we drop it and write P and P_c.

We define an *(α, β)-approximate packing* as one where: (1) $P \leqslant \alpha\mathrm{OPT}(I) + O(1)$ and (2) for each colour $c \in C$, $P_c \leqslant \beta\mathrm{OPT}(I_c) + O(1)$. An algorithm that always produces (α, β)-approximate packings is called an *(α, β)-approximation algorithm*.

As usual in bin packing problems, we allow additive constants and consider α (respectively β) as the asymptotic performance ratio as $\mathrm{OPT}(I)$ (respectively $\mathrm{OPT}(I_c)$) grows to infinity (and hence the total weight of items). This is because a simple reduction from PARTITION (e.g. see [10]) shows that, without allowing additive constants, it would be NP-hard to do better than $(1.5 - \varepsilon, \delta)$ or $(\delta, 1.5 - \varepsilon)$ approximate packings for any δ.

When dealing with the online problem we have similar definitions for the competitive ratio of an online algorithm, and in this case $\mathrm{OPT}(I)$ corresponds to an optimal offline solution to instance I that has full knowledge of the request sequence I. As standard, we shall use the term approximation ratio interchangeably with competitive ratio when discussing online algorithms (i.e. a 2-approximate online scheme is one that is within a factor 2 of the optimal offline scheme).

2.1. Related work

Chung et al. [4] consider the case where each item is of a different colour and can be fractionally (arbitrarily) divided between bins, bins have different sizes and the total weight of items exactly equals the total weight of bins. They show how to compute an allocation that is asymptotically optimal for each colour. By contrast, we relax the assumption that we must exactly fill all the bins, and consider the case of indivisible allocations. In this setting, the problem is much more interesting: it is impossible to get arbitrarily good $(1 + \varepsilon, 1 + \varepsilon)$-approximate allocations in general. Thus, these relaxed packings have a tradeoff between bin stretch and colour stretch, with polynomial-time approximations. We also consider for the first time the case where items arrive online. However, the case of heterogeneous bins is open for our setting.

The nonexpansive hashing scheme of Linial and Sasson [16] can also be used to find a locality-preserving packing for unit-size items. By defining the distance of two items to be 0 if they are of the same colour, and $\delta > 1$ otherwise, one can interpret their dynamic hashing result as follows: for any $\varepsilon > 0$, it is possible to hash unit-size items into bins in $O(1)$ time so that they have use $O(\mathrm{OPT}^{1+\varepsilon})$ bins (giving bin stretch $O(\mathrm{OPT}^{\varepsilon})$ and colour stretch $O(1)$).

Krumke et al. [14] study a related 'online coloured bin packing' problem where the goal is to minimize the number of different colours packed into each bin, while using the entire capacity of each bin (in their problem all items have same unit size). However, this problem is quite different to ours. In particular, an optimal solution problem when minimizing the number of colours per bin may give arbitrarily bad bin stretch. Consider b bins of capacity x, and unit size items of many colours $c_1, c_2, \ldots, c_{(x-2)b+1}$. There will be $2b$ items of colour c_1 and 1 item of each of the other colours. Now, a $(1, 1)$-approximate packing places x colours from $\{c_2 \ldots c_{(x-2)b+1}\}$ into each bin and the items of c_1 into the remaining bins. On the other hand, a packing minimizing the maximum number of colours per bin (while using all the capacity of each