On Multi-dimensional Packing Problems*

Chandra Chekuri[†] Sanjeev Khanna[‡]

Abstract

We study the approximability of multi-dimensional generalizations of three classical packing problems: multiprocessor scheduling, bin packing, and the knapsack problem. Specifically, we study the *vector scheduling* problem, its dual problem, namely, the *vector bin packing* problem, and a class of *packing integer programs*. The vector scheduling problem is to schedule n d-dimensional tasks on m machines such that the maximum load over all dimensions and all machines is minimized. The vector bin packing problem, on the other hand, seeks to minimize the number of bins needed to schedule all n tasks such that the maximum load on any dimension across all bins is bounded by a fixed quantity, say 1. Such problems naturally arise when scheduling tasks that have multiple resource requirements. Finally, packing integer programs capture a core problem that directly relates to both vector scheduling and vector bin packing, namely, the problem of packing a maximum number of vectors in a single bin of unit height. We obtain a variety of new algorithmic as well as inapproximability results for these three problems.

Keywords : Multi-dimensional packing, vector scheduling, vector bin packing, packing integer programs, multiprocessor scheduling, bin packing, knapsack, approximation algorithms, hardness of approximation, combinatorial optimization.

AMS Subject Classification : 68Q25.

*A preliminary version of this paper appeared in the *Proceedings of the 10th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 185–194, 1999.

[†]Bell Labs, 600-700 Mountain Ave, Murray Hill, NJ 07974. E-mail: `chekuri@research.bell-labs.com`. Most of this work was done during a summer internship at Bell Labs while the author was at Stanford University. Work at Stanford was supported by an IBM Co-operative fellowship, an ARO MURI Grant DAAH04-96-1-0007, and NSF Award CCR-9357849.

[‡]Dept. of CIS, University of Pennsylvania, Philadelphia, PA 19104. E-mail: `sanjeev@cis.upenn.edu`. This work was done when this author was at Bell Labs, Lucent Technologies.

1 Introduction

Multi-processor scheduling, bin packing, and the knapsack problem are very well studied problems in combinatorial optimization. Their study has had a large impact on the design and analysis of approximation algorithms. All of these problems involve packing items of different sizes into bins of finite capacities. In this work we study *multi-dimensional* generalizations of these problems where the items to be packed are d-dimensional vectors and bins are d-dimensional objects as well. We obtain a variety of approximability and inapproximability results, in the process, significantly improving upon earlier known results for these problems. Some of our results include a polynomial time approximation scheme (PTAS) for the vector scheduling problem when the dimension is fixed, and an approximation algorithm for the vector bin-packing problem that improves a two-decade old bound. Though our primary motivation is vector scheduling and vector bin packing, an underlying problem that arises is the problem of maximizing the numbers of vectors that can be packed into a bin of fixed size. This is a special case of the multi-dimensional knapsack problem that is equivalent to packing integer programs (PIPs) [27, 29]. PIPs are an important class of integer programs that capture several NP-hard combinatorial optimization problems including the maximum independent set problem, the disjoint paths problem, and hypergraph matchings. The only general technique known for approximating PIPs is to use randomized rounding on the natural LP relaxation [27, 29]. We show here that the approximation guarantee for PIPs, as obtained via randomized rounding, is essentially the best possible unless NP=ZPP.

In addition to their theoretical importance, these problems have several applications such as load balancing, cutting stock, and resource allocation, to name a few. One of our motivations for studying these problems comes from recent interest [12, 13, 14] in multi-dimensional resource scheduling problems in parallel query optimization. A favored architecture for parallel databases is the so called shared-nothing environment [5] where the parallel system consists of a set of independent processing units each of which has a set of time-sharable resources such as CPU, one or more disks, network controllers etc. A task executing on one of these units has requirements from each of these resources and is best described as a multi-dimensional load vector. However in most work on scheduling, both in theory and practice, it is assumed that the load of a task is described by a single aggregate work measure. This simplification is done typically to reduce the complexity of the scheduling problem. However, for large task systems that are typically encountered in database applications, ignoring the multi-dimensionality could lead to bad performance. The work in [11, 12, 13, 14] demonstrates the practical effectiveness of the multi-dimensional approach. One of the basic resource scheduling problem that is considered in the above papers is the problem of scheduling d-dimensional vectors (tasks) on d-dimensional bins (machines) to minimize the maximum load on any dimension (the load on the most loaded resource). Surprisingly, despite the large body of work on approximation algorithms for multi-processor scheduling and its several variants [15, 23], the authors in [11] had to settle for a naive $(d+1)$-approximation for the d-dimensional vector scheduling problem. Our work here provides a PTAS when d is fixed and an $O(\ln^2 d)$-approximation when d is arbitrary. A similar situation existed for the vector bin packing problem where the best known approximation ratio prior to our work was $(d+\epsilon)$. In this paper, we improve this to obtain a $(1+\epsilon \cdot d + O(\ln \epsilon^{-1}))$-approximation for any fixed $\epsilon > 0$. In what follows, we formally define the problems that we study and provide a detailed description of our results.

1.1 Problem Definitions

We start by defining the vector scheduling problem. For a vector x, the quantity $\|x\|_\infty$ denotes the standard ℓ_∞ norm.

Definition 1.1 (Vector Scheduling (VS)) *We are given a set J of n rational d-dimensional vectors $p_1, \ldots, p_n$ from $[0,\infty)^d$ and a number m. A valid solution is a partition of J into m sets $A_1, \ldots, A_m$. The objective is to minimize $\max_{1 \le i \le m} \|\bar{A}_i\|_\infty$ where $\bar{A}_i = \sum_{j \in A_i} p_j$ is the sum of the vectors in A_i.*

Definition 1.2 (Vector Bin Packing (VBP)) *Given a set of n rational vectors $p_1, \ldots, p_n$ from $[0,1]^d$, find a partition of the set into sets $A_1, \ldots, A_m$ such that $\|\bar{A}_i\|_\infty \le 1$ for $1 \le j \le m$. The objective is to minimize m, the size of the partition.*

The following definition of PIPs is from [29]. In the literature this problem is also referred to as the d-dimensional 0-1 knapsack problem [7].

Definition 1.3 (Packing Integer Program (PIP)) *Given $A \in [0,1]^{d \times n}$, $b \in [1,\infty)^d$, and $c \in [0,1]^n$ with $\max_j c_j = 1$, a packing integer program (PIP) seeks to maximize $c^T x$ subject to $x \in \{0,1\}^n$ and $Ax \le b$. Furthermore if $A \in \{0,1\}^{d \times n}$, b is assumed to be integral. Finally B is defined to be $\min_i b_i$.*

The restrictions on A, b, and c in the above definition are without loss of generality: an arbitrary packing problem can be reduced to the above form (see [29]). We are interested in PIPs where $b_i = B$ for $1 \le i \le d$. When $A \in \{0,1\}^{d \times n}$ this problem is known as the *simple B-matching in hypergraphs* [24]: given a hypergraph with non-negative edge weights, find a maximum weight collection of edges such that no vertex occurs in more than B of them. When $B = 1$ this is the usual hypergraph matching problem. We note that the maximum *independent set* problem in graphs is a special case of the hypergraph matching problem.

1.2 Related Work and Our Results

All the problems we consider are NP-Complete for $d = 1$ (multi-processor scheduling, bin packing, and the knapsack problem). The dimension of the vectors, d, plays an important role in determining the complexity. We concentrate on two cases, when d is fixed constant, and when d is part of the input and can be arbitrary. Below is an outline of the various positive and negative results that we obtain for these problems.

Vector Scheduling: For the vector scheduling problem the best approximation algorithm [13] prior to our work had a ratio of $(d+1)$. When d is a fixed constant (a case of practical interest) we obtain a polynomial time approximation scheme (PTAS), generalizing the result of Hochbaum and Shmoys [19] for multiprocessor scheduling. In addition we obtain a simpler $O(\ln d)$-approximation algorithm that is better than $(d+1)$ for all $d \geq 2$. When d is large we give an $O(\ln^2 d)$-approximation that uses known approximation algorithms for PIPs as a subroutine. We also give a very simple $O(\ln dm / \ln \ln dm)$-approximation. Finally, we show that it is hard to approximate the VS problem to within *any* constant factor when d is arbitrary.

Vector Bin Packing: The previous best known approximation algorithms for this problem gave a ratio of $(d + \epsilon)$ for any fixed $\epsilon > 0$ [4] and $(d + 7/10)$ [9]; the latter result holds even in an online setting. All the ratios mentioned are asymptotic, that is there is an additive term depending on d and on ϵ. Karp et al. [22] do a probabilistic analysis and show bounds on the average wastage in the bins. We design an approximation algorithm that for any fixed $\epsilon > 0$, achieves a $(1+\epsilon \cdot d+O(\ln \epsilon^{-1}))$-approximation in polynomial time, thus improving upon the previous guarantees. One useful corollary of this result is that when d is a fixed constant we can approximate the problem to within a ratio of $O(\ln d)$. When d is arbitrary a simple reduction from the graph coloring problem gives a $d^{\frac{1}{2}-\epsilon}$ hardness for any fixed $\epsilon > 0$ even when vectors are drawn from the set $[0,1]^d$. Moreover, even when $d = 2$ the problem is APX-hard [31];an interesting departure from classical bin packing problem ($d = 1$) which exhibits an asymptotic FPTAS.

Packing Integer Programs: For fixed d there is a PTAS for PIPs [7]. For large d the randomized rounding technique of Raghavan and Thompson [27] yields integral solutions of value $t_1 = \Omega(\mathrm{OPT}/d^{1/B})$ if $A \in [0,1]^{d \times n}$, and $t_2 = \Omega(\mathrm{OPT}/d^{1/(B+1)})$ if $A \in \{0,1\}^{d \times n}$. Srinivasan [29] improved these results to obtain solutions of value $\Omega(t_1^{B/(B-1)})$ and $\Omega(t_2^{(B+1)/B})$ respectively (see discussion at the end of Section 4.1 concerning when these values are better). Thus the parameter B plays an important role in the approximation ratio achieved, with better ratios obtained as B gets larger (recall that entries in A are upper bounded by 1). It is natural to question if the dependence of the approximation ratio on B could be any better. We show that PIPs are hard to approximate to within a factor of $\Omega(d^{\frac{1}{B+1}-\epsilon})$ for every fixed B, thus establishing that randomized rounding essentially gives the best possible approximation guarantees. Hardness was known only for the case $B = 1$ via a reduction from the maximum independent set problem. We show how this can amplified to work for larger values of B and then use Håstad's result [18] on the inapproximability of the maximum independent set problem. An interesting aspect of our reduction is that the hardness result holds even when the optimal is restricted to choosing a solution that satisfies $Ax \leq 1^d$ while the approximation algorithm is only required to satisfy the relaxed constraint of $Ax \leq B^d$.

Table 1 summarizes our results. For conciseness, in the table below when we indicate that a problem is c-hard we mean that, unless NP = ZPP, no polynomial time algorithm can approximate it to within a factor of c.

Problem	$d = 1$	Constant $d \geq 2$	Arbitrary d
Vector Scheduling	PTAS [19]	$(d+1)$ (folklore) PTAS (this paper)	$(d+1)$ (folklore) $O(\ln^2 d)$ (this paper)
	NP-hard	NP-hard	NP-hard c-hard $\forall\ c > 1$ (this paper)
Vector Bin Packing	APTAS[1] [4, 21]	$(d+\epsilon)$ [4] $O(\ln d)$ (this paper)	$(d+\epsilon)$ [4] $1+\epsilon d+O(\ln\frac{1}{\epsilon})$ (this paper)
	NP-hard	APX-hard [31]	APX-hard [31] $d^{\frac{1}{2}-\epsilon}$-hard (this paper)
Packing Integer Programs	FPTAS [20]	PTAS [7]	$O(d^{\frac{1}{(B+1)}})$ [26, 27, 29]
	NP-hard	NP-hard	$d^{\frac{1}{2}-\epsilon}$-hard for $B = 1$ $d^{\frac{1}{(B+1)}-\epsilon}$-hard $\forall\ B \geq 1$ (this paper)

Table 1: Approximation bounds and inapproximability results for each problem.

1.3 Organization

The rest of the paper is organized as follows. Sections 2 and 3 present our approximation algorithms for the vector scheduling problem and the vector bin packing problem respectively. In Section 4 we

[1]APTAS denotes an *asymptotic* PTAS. A problem has an APTAS if for any fixed $\epsilon > 0$, there exists a positive integer N_ϵ such that there is a polynomial time $(1+\epsilon)$-approximation algorithm for all instances of the problem with optimum value at least N_ϵ.

present our inapproximability results for the three problems.

2 Algorithms for Vector Scheduling

For any set of vectors (jobs) A, we define $\bar{A}$ to be the vector sum $\sum_{j \in A} p_j$. The quantity $\bar{A}^i$ denotes component i of the vector $\bar{A}$. Throughout this section, we assume without loss of generality that the vectors have been scaled such that the optimal schedule value is 1.

2.1 A PTAS for fixed d

Hochbaum and Shmoys [19] gave a PTAS for the multi-processor scheduling problem (VS problem with $d = 1$) using dual approximation schemes. We now show that a non-trivial generalization of their ideas yields a PTAS for arbitrary but fixed d.

The basic idea used in [19] is a primal-dual approach whereby the scheduling problem is viewed as a bin packing problem. If an optimal solution can pack all jobs with load not exceeding some height h, assume $h = 1$ from here on, then the scheduling problem is to pack all the jobs into m bins (machines) of height 1. The authors then give an algorithm to solve this bin packing problem with bin height relaxed to $(1 + \epsilon)$ where $\epsilon > 0$ is a fixed constant. In order to do so, they classify a job as large or small depending on whether its size is greater than ϵ or not. Only a fixed number (at most $1/\epsilon$) of large jobs can be packed into any bin. The sizes of the large jobs are then rounded up to be one of $O(\ln 1/\epsilon)$ distinct values. Dynamic programming is used to pack the rounded up large jobs into the m bins such that no bin exceeds a height of $(1 + \epsilon)$. The small jobs are then greedily packed on top of the large jobs.

We take a similar approach to the problem. Our dual problem is vector bin packing. The primary difficulty in generalizing the above ideas to the case of jobs or vectors of $d \geq 2$ dimensions is the lack of a total order on the "size" of the jobs. It is still possible to classify a vectors as large or small depending on its ℓ_∞ norm. However, the scheme of [19] whereby the small jobs are greedily packed on top of the large jobs does not apply. We need to take into account the interaction between the packing of large and small vectors. In addition, the packing of small vectors is non-trivial. In fact we use a linear programming relaxation and a careful rounding to pack the small jobs. We describe our ideas in detail below. Following the above discussion we will think of machines as d-dimensional bins and the schedule length as bin capacity (height). Given an $\epsilon > 0$ and a guess for the optimal value (that we assume is normalized to 1), we describe an ϵ-relaxed decision procedure A_ϵ that either returns a schedule of height $(1 + \epsilon)$ or proves that the guess is incorrect. We can use A_ϵ to do a binary search for the optimal value. Let δ be ϵ/d.

Preprocessing Step: Our first idea is to reduce to zero all coordinates of the vectors that are too small relative to the largest coordinate. This allows us to bound the ratio of the largest coordinate to the smallest non-zero coordinate.

Lemma 2.1 *Let I be an instance of the VS problem. Let I' be a modified instance where we replace each p_j in I with a vector q_j as follows. For each $1 \le i \le d$, $q_j^i = p_j^i$ if $p_j^i > \delta \|p_j\|_\infty$ and $q_j^i = 0$ otherwise. Then replacing the vector q_j by the vector p_j in any valid solution to I' results in a valid solution to I of height at most a factor of $(1+\epsilon)$ that of I'.*

Proof. Let A be a set of vectors in I and B the corresponding set of vectors in I'. Then it follows from the transformation described above that

$$\begin{aligned}
\bar{A}^i &\le \bar{B}^i + \delta \sum_{j \in B} \|q_j\|_\infty \\
&\le \bar{B}^i + \delta \sum_{j \in B} \|q_j\|_1 \\
&\le \bar{B}^i + \delta \|B\|_1 \\
&\le \bar{B}^i + \delta d \|B\|_\infty \\
&\le \bar{B}^i + \epsilon \|B\|_\infty .
\end{aligned}$$

Therefore $\|A\|_\infty \le (1+\epsilon)\|B\|_\infty$. It follows that replacing vectors in I' by those in I' increases the height of the machines by only a $(1+\epsilon)$ factor. □

Large versus Small Vectors: Assume from here on that we have transformed our instance as described in the above lemma. The second step in the algorithm is to partition the vectors into two sets L and S corresponding to *large* and *small*. L consists of all vectors whose ℓ_∞ norm is greater than δ, and S is the rest of the vectors. The algorithm A_ϵ will have two stages; the first stage packs all the large jobs, and the second stage packs the small jobs. Unlike the case of $d = 1$, the interaction between the two stages has to be taken into account for $d \ge 2$. We show that the interaction can be captured in a compact way as follows. Let $(a_1, a_2, \ldots, a_d)$ be a d-tuple of integers such that $0 \le a_i \le \lceil 1/\epsilon \rceil$. We will call each such distinct tuple a *capacity configuration*. There are at most $t = (1 + \lceil 1/\epsilon \rceil)^d$ such configurations. Assume that the t capacity configurations (tuples) are ordered in some way and let a_i^k be the value of coordinate i in tuple k. A capacity configuration approximately describes how a bin is filled. However we have m bins. A t-tuple $(m_1, \ldots, m_t)$ where $0 \le m_i \le m$ and $\sum_i m_i = m$ is called a *bin configuration* that describes the number of bins of each capacity configuration. The number of possible bin configurations is clearly $O(m^t)$. Since there are only a polynomial number of such configurations for fixed d and ϵ we can "guess" the configuration used by a feasible packing. A packing of vectors in a bin is said to *respect* a capacity configuration $(a_1, \ldots, a_d)$ if the height of the packing in each dimension i is less than ϵa_i. Given a capacity configuration we can define the corresponding *empty capacity configuration* as the tuple obtained by subtracting each entry from $(\lceil 1/\epsilon \rceil + 1)$. For a bin configuration M we denote by $\bar{M}$ the corresponding bin configuration as the one obtained by taking the empty capacity configurations for each of the bins in M.

Overview of the Algorithm: The algorithm performs the following steps for each bin configuration M:

(a) decide if vectors in L can be packed respecting M.

(b) decide if vectors in S can be packed respecting $\bar{M}$.

If both steps above succeed for some M, we have a packing of height at most $(1+\epsilon)$. Otherwise we will prove that the assumption that the optimal packing has a height of 1 is false.

Packing the large vectors: The first stage consists of packing the vectors in L. Observe that the smallest non-zero coordinate of the vectors in L is at least δ^2. We partition the interval $[\delta^2, 1]$ into $q = \lceil \frac{2}{\epsilon} \ln \delta^{-1} \rceil$ intervals of the form $(x_0, (1+\epsilon)x_0], (x_1, (1+\epsilon)x_1], \ldots, (x_{q-1}, 1]$ where $x_0 = \delta^2$ and $x_{i+1} = (1+\epsilon)x_i$. We discretize every non-zero coordinate of the vectors in L by rounding the coordinate down to the left end point of the interval in which it falls. Let L' be the resulting set of vectors.

Lemma 2.2 *Let I' be an instance obtained from the original instance I by rounding vectors in L as described above. Then replacing each vector in L' by the corresponding vector in L in any solution for I' results in a solution for I of height at most $(1+\epsilon)$ times that of I'.*

Proof. Each coordinate of a vector in L' is at least $(1+\epsilon)^{-1}$ times the coordinate of the corresponding vector in L. The lemma follows trivially. □

Vectors in L' can be classified into one of $s = (1 + \lceil \frac{2}{\epsilon} \ln \delta^{-1} \rceil)^d$ distinct classes. Any packing of the vectors into one bin can be described as a tuple $(k_1, k_2, \ldots, k_s)$ where k_i indicates the number of vectors of the ith class. Note that at most d/δ vectors from L' can be packed in any bin. Therefore $\sum k_i \leq d/\delta$. Thus there are at most $(d/\delta)^s$ *configurations*. A configuration is *feasible* for a capacity configuration if the vectors described by the configuration can be packed without violating the height constraints described by the capacity configuration. Let C_k denote the set of all configurations of the jobs in L' that are feasible for the kth capacity configuration. From our discussion $|C_k| \leq (d/\delta)^s$.

Lemma 2.3 *Let* $M = (m_1, m_2, \ldots, m_t)$ *be a bin configuration. There exists an algorithm with running time* $O((d/\delta)^s mn^s)$ *to decide if there is a packing of the jobs in* L' *that respects* M*.*

Proof. We use a simple dynamic programming based algorithm. Observe that the number of vector classes in L' is at most s. Thus any subset of vectors from L' can be specified by a tuple of size s and there are $O(n^s)$ distinct tuples. The algorithm orders bins in some arbitrary way and to each bin assigns a capacity configuration from M. For $1 \leq i \leq m$, the algorithm computes all possible subsets of vectors from L' (tuples) that can be packed validly in the first i bins. For each i this information can be maintained in $O(n^s)$ space. Given the tuples for bin i, the tuples for bin $(i+1)$ can be computed in $O(d/\delta)^s$ time per tuple since that is an upper bound on the number of feasible configurations for any capacity configuration. Thus for each bin i, in $O(d/\delta)^s n^s)$ time, we can compute the tuples that can packed into the first i bins given the information for bin $(i-1)$. The number of bins is m so we get the required time bound. □

Packing the small vectors: We now describe the second stage, that of packing the vectors in S. For the second stage we write an integer programming formulation and round the resulting LP relaxation to find an approximate feasible solution. Without loss of generality assume that the vectors in S are numbered 1 to $|S|$. The IP formulation has 0-1 variables x_{ij} for $1 \leq i \leq |S|$ and $1 \leq j \leq m$. Variable x_{ij} is 1 if p_i is assigned to machine j. Every vector has to be assigned to some machine. This results in the following equation.

$$\sum_j x_{ij} = 1 \qquad\qquad 1 \leq i \leq |S| \tag{1}$$

Given a bin configuration M we can define for each machine j and dimension k a height bound b_j^k that an assignment should satisfy. Thus we obtain

$$\sum_i p_i^k \cdot x_{ij} \leq b_j^k \qquad 1 \leq j \leq m, 1 \leq k \leq d. \tag{2}$$

In addition we have the integrality constraints, namely, $x_{ij} \in \{0,1\}$. We obtain a linear program by replacing these constraints by the following.

$$x_{ij} \geq 0 \tag{3}$$

Proposition 1 *Any* basic feasible solution *to the LP defined by Equations 1, 2, and 3 has at most* $d \cdot m$ *vectors that are assigned fractionally to more than one machine.*

Proof. The number of variables in our LP is $n \cdot m$. The number of non-trivial constraints (those that are other than $x_{ij} \geq 0$) is $(n + d \cdot m)$. From standard polyhedral theory [28] any basic (vertex) solution to our LP has $n \cdot m$ tight constraints. Therefore by a simple counting argument at most

$(n + d \cdot m)$ variables can be strictly positive. Since each vector is assigned to at least one machine, the number of vectors that are fractionally assigned to more than one machine is at most $d \cdot m$. $\square$

We can solve the above linear program in polynomial time and obtain a basic feasible solution. Let S' be the set of vectors that are not assigned integrally to any machine. By the above lemma $|S'| \leq d \cdot m$. We partition the set S' into m sets of at most d vectors each in an *arbitrary* manner and assign the ith set to the ith machine. Since $\|p_j\|_\infty \leq \delta = \epsilon/d$ for every $j \in S'$, the above step does not violate the height by more than ϵ in any dimension.

Putting together all the ingredients we obtain our main theorem below.

Theorem 1 *Given any fixed $\epsilon > 0$, there is a $(1+\epsilon)$ approximation algorithm for the VS problem that runs in $(nd/\epsilon)^{O(s)}$ time where $s = O((\frac{\ln(d/\epsilon)}{\epsilon})^d)$.*

Proof. Given a correct guess for the optimal schedule height it is clear from the description that we obtain a $(1 + O(\epsilon))$-approximation. Following the overview of the algorithm we find a packing of vectors in L and S for each choice of bin configuration M. The running time is dominated by the time to pack L. Since there are at most $m^t = O(n^{O(\epsilon^{-d})})$ bin configurations the running time follows from Lemma 2.3. We can guess the optimal value to within a $(1+\epsilon)$ precision using $O(\ln d/\epsilon)$ guesses using a simple $(d+1)$-approximation described in Subsection 2.2. $\square$

2.2 The General Case

We now consider the case when d is arbitrary and present two approximation algorithms for this case. The first algorithm is deterministic and has an approximation ratio that is only a function of d ($O(\ln^2 d)$) while the second algorithm is randomized and achieves an approximation ratio that is a function of both d and m ($O(\ln dm/\ln\ln dm)$). Given a set of positive vectors A we denote by $\mathcal{V}(A)$ the *volume* of A which is the sum of all coordinates of all vectors in A, in other words the ℓ_1 norm of $\sum_{j \in A} p_j$. We once again assume that the optimal schedule height is 1.

2.2.1 An $O(\ln^2 d)$-approximation

We start by analyzing a simple algorithm which will serve as a base case for our $O(\ln^2 d)$-approximation algorithm. Recall that J is the set of vectors in the input instance. The infinity norm of each of the job vectors is clearly a lower bound on the optimal value. Hence

$$\max_{j \in J} \|p_j\|_\infty \leq 1. \tag{4}$$

The second lower bound is obtained by using the average volume per dimension.

$$\frac{\mathcal{V}(J)}{m \cdot d} \leq 1. \tag{5}$$

We can strengthen the above bound by splitting the sum dimension wise.

$$\max_{i=1}^{d} \frac{\bar{J}^i}{m} \leq 1. \tag{6}$$

A naive algorithm for our problem is to ignore the multi-dimensional aspect of the jobs and treat them as a one dimensional vectors of size equal to the sum of their components. The dimensionality of the bins is also ignored. Then one can apply the standard list scheduling algorithm of Graham [16] for multiprocessor scheduling to obtain the following theorem that uses the simple lower bounds developed above.

Lemma 2.4 *Applying list scheduling on the volumes of the vectors results in a schedule of height at most* $\frac{\mathcal{V}(J)}{m} + \max_{j \in J} \|p_j\|_\infty$. *This yields a* $(d+1)$*-approximation.*

Proof. The upper bound follows from standard analysis of list scheduling. The approximation guarantee follows from the lower bounds in Equations 4 and 5. □

The $O(\ln^2 d)$-approximation Algorithm: Before we state the algorithm formally we need a couple of definitions. The following problem is a special case of a general PIP.

Definition 2.1 *Given a set* J *of* n *vectors in* $[0,1]^d$, *the* largest volume packing problem *is the problem of finding a subset* S *such that* $\|\bar{S}\|_\infty \leq 1$ *and* $\mathcal{V}(S)$ *is maximized. Let* $\mathcal{V}_{\max}$ *denote the value of the optimal solution.*

Definition 2.2 *An* (α, β)*-approximation to the largest volume packing problem is a subset* S *that satisfies the conditions* $\|\bar{S}\|_\infty \leq \alpha$ *and* $\mathcal{V}(S) \geq \beta \mathcal{V}_{\max}$.

We will typically use the above definition with $\alpha \geq 1$ and $\beta \leq 1$.

Algorithm Volume Pack:

1. **repeat** for t stages
   (a) **for** $k = 1$ to m **do**
      i. Find an (α, β)-approximation to the largest volume packing problem with the current set of job vectors.
      ii. Allocate jobs in packing to machine k and remove them.
2. Find a separate schedule for the remaining jobs using naive volume based list scheduling.
3. Combine the two schedules machine by machine in the obvious way.

We now prove several simple lemmas to analyze the performance of the above algorithm.

Lemma 2.5 *Let $J(i)$ be the set of jobs remaining at the beginning of the ith stage with $J(1) = J$. Let $J_k(i)$ be the set of jobs remaining after machine k has been packed in stage i. Then*

$$\mathcal{V}(J_k(i)) \leq \mathcal{V}(J(i)) \cdot (1 - \beta/m)^k$$

Proof. We prove the lemma by induction on k. The claim is trivially true for $k = 0$. Suppose the claim is true up to machine k. We show that the claim is true for machine $(k+1)$. Since all jobs in J can be scheduled on m machines with height 1 it follows that all jobs in $J_k(i)$ can be likewise scheduled. By a simple averaging argument we can infer that there exists a set of jobs in $J_k(i)$ with volume at least $\mathcal{V}(J_k(i))/m$ that can be packed in a machine with height at most 1. Since we obtain a β -approximation to largest volume packing, we pack jobs with a volume of at least $\beta \cdot \mathcal{V}(J_k(i))/m$. Therefore $\mathcal{V}(J_{(k+1)}(i)) \leq \mathcal{V}(J_k(i)) \cdot (1 - \beta/m)$. By our induction hypothesis $\mathcal{V}(J_k(i)) \leq \mathcal{V}(J(i)) \cdot (1 - \beta/m)^k$. The lemma follows. □

Corollary 1 $\mathcal{V}(J(i)) \leq \mathcal{V}(J)/e^{(i-1)\beta}$.

Proof. From Lemma 2.5 $\mathcal{V}(J(i)) \leq \mathcal{V}(J(i-1)) \cdot (1 - \beta/m)^m$. Since $(1 - \beta/m)^m \leq e^{-\beta}$ we get the required bound. □

Lemma 2.6 *Algorithm* Volume Pack *yields a schedule of height at most* $(t \cdot \alpha + \frac{d}{e^{t\beta}} + 1)$.

Proof. Consider the machine that achieves the maximum height in the schedule produced by Volume Pack. Let J_1 and J_2 be the set of jobs allocated to that machine in the packing stage and the list scheduling stage respectively. From the packing property it is easy to see that the height of machine due to jobs in J_1 is at most $t\alpha$. Let J' be the set of jobs remaining after the t stages of packing. These are scheduled using list scheduling. From Corollary 1 we have that

$$\mathcal{V}(J') \leq \mathcal{V}(J)/e^{t\beta}.$$

From Lemma 2.4, the height increase of the machine due to jobs in J_2 is at most

$$\mathcal{V}(J')/m + \max_j \|p_j\|_\infty \leq \frac{d}{e^{t\beta}} \cdot \mathcal{V}(J)/(dm) + 1 \leq \frac{d}{e^{t\beta}} + 1.$$

In the above inequality we are using the fact that the two lower bounds are less than 1, the optimal value. Combining the two equations gives us the desired bound. □

The parameter t in the algorithm can be chosen as a function of α and β to obtain the best ratio. Note that the largest volume packing problem is a special case of a PIP where c_i is simply the volume of vector i. PIPs have a $(O(\ln d), 1/2)$-approximation via randomized rounding [27, 29] that can be derandomized by techniques from [26]. When d is fixed there is a $(1, 1-\epsilon)$ approximation [7] that runs in time polynomial in $n^{d/\epsilon}$. These observations imply the following.

Theorem 2 *There is an $O(\ln^2 d)$-approximation algorithm for the VS problem.*

Theorem 3 *There is an $O(\ln d)$-approximation algorithm for the VS problem that runs in time polynomial in n^d.*

2.2.2 An $O(\ln dm/\ln\ln dm)$-Approximation

The approximation result of Theorem 2 is good when d is small compared to m. However when d is large we can obtain a $O(\ln dm/\ln\ln dm)$-approximation by a simple randomized algorithm that assigns each vector independently to a machine chosen uniformly at random from the set of m machines. Theorem 4 bounds the performance of this algorithm, referred to as *Random*. We need a version of the standard Chernoff bound on sums of independent random variables.

Proposition 2 *Let $X_1, \ldots, X_n$ be independent binary random variables and let $X = \sum_{i=1}^n t_i X_i$. Let $T = \max_i t_i$ and $\mu = \mathbf{E}[X]$. Then, for any sufficiently large h, $\mathbf{Pr}\left[X > (1 + c\frac{\ln h}{\ln\ln h})(\mu + T)\right] \leq h^{-c/2}$.*

Proof. Let Y_i be a random variable such that $Y_i = \frac{t_i}{T} X_i$. Note that Y_i takes on values in $[0,1]$. Let $Y = \sum_i Y_i$. It follows that $Y = X/T$ and that $\mathbf{E}[Y] = \mu/T$. Therefore, for any $\delta > 0$, $\mathbf{Pr}[X > (1+\delta)(\mu + T)] = \mathbf{Pr}[Y > (1+\delta)(\mu/T + 1)]$. Applying the standard Chernoff-Hoeffding bounds [25] on sums of independent random variables that assume values in $[0,1]$ to Y, we get the desired result. $\square$

Theorem 4 *Random gives an $O(\ln dm/\ln\ln dm)$-approximation with high probability.*

Proof. Consider the first machine. Let X_j be the indicator random variable that is 1 if vector j is assigned to the first machine. The X_j's are independent. By uniformity $\mathbf{Pr}[X_j = 1] = 1/m$. Let $P = \sum_j p_j X_j$. Note that P is a vector since each p_j is a vector: let P^i denote the ith coordinate of P. By linearity of expectations $\mathbf{E}[P^i] = \sum_j p_j^i/m \leq 1$ (using Equation 5). Also observe that $\max_j p_j^i \leq 1$ (using Equation 4). Now we estimate the probability that P^i deviates significantly from its expected value. From Proposition 2, $\mathbf{Pr}\left[P^i > (\mathbf{E}[P^i] + \max_j p_j^i)(1 + c\ln dm/\ln\ln dm)\right] \leq (dm)^{-c/2}$. Thus with high probability P^i is $O(\ln dm/\ln\ln dm)$. In general, if A_k^i is the event that the ith dimension of machine k is greater than $2(1+c)\ln dm/\ln\ln dm$, then from above we know that $\mathbf{Pr}[A_k^i] \leq (dm)^{-c/2}$. Thus $\mathbf{Pr}\left[(A = \cup_{i=1}^d \cup_{k=1}^m A_k^i)\right] \leq dm(dm)^{-c/2}$. By choosing c sufficiently large we can ensure that $\mathbf{Pr}[A]$ is less than an inverse polynomial factor. But the complement of A is the event that the schedule length is $O(\ln dm/\ln\ln dm)$. Thus with high probability we get an $O(\ln dm/\ln\ln dm)$-approximation. $\square$

3 Algorithms for Vector Bin Packing

We now examine the problem of packing a given set of vectors into the smallest possible number of bins. Our main result here is as follows:

Theorem 5 *For any fixed $\epsilon > 0$, we can obtain in polynomial time a $(1 + \epsilon \cdot d + O(\ln(1/\epsilon)))$-approximate solution for vector bin packing.*

This improves upon the long standing $(d + \epsilon)$-approximation algorithm of [4]. Our approach is based on solving a linear programming relaxation for this problem. As in Section 2.1, we use a variable x_{ij} to indicate if vector p_i is assigned to bin j. We guess the least number of bins m (easily located via binary search) for which the following LP relaxation is feasible; clearly $m \leq \text{OPT}$.

$$
\begin{aligned}
\sum_j x_{ij} &= 1 && 1 \leq i \leq n \\
\sum_i p_i^k \cdot x_{ij} &\leq 1 && 1 \leq j \leq m \ , \ 1 \leq k \leq d \\
x_{ij} &\geq 0 && 1 \leq i \leq n \ , \ 1 \leq j \leq m
\end{aligned}
$$

Once again, we use the fact that a basic feasible solution would make fractional bin assignments for at most $d \cdot m$ vectors. Thus at this point, all but a set S of at most $d \cdot m$ vectors have integral assignments in m bins. To find a bin assignment for S, we repeatedly find a set $S' \subseteq S$ of up to $k = \lceil 1/\epsilon \rceil$ vectors that can all be packed together and assign them to a new bin. This step is performed greedily, i.e. we seek to find a largest possible such set in each iteration. We can perform this step by trying out all possible sets of vectors of cardinality less than $(k+1)$. We now claim that this procedure must terminate in $\epsilon \cdot d \cdot m + O(\ln \epsilon^{-1}) \cdot \text{OPT}$ steps. To see this, consider the first time that we pack less than k vectors in a bin. The number of bins used thus far is bounded by $(d \cdot m)/k$. Moreover, the total number of vectors that remain at this point is at most $(k-1)\text{OPT}$; let S' denote this remaining set of vectors. Since the optimal algorithm cannot pack more than $(k-1)$ vectors of S' in one bin, our greedy bin assignment procedure is identical to a greedy set cover algorithm where each set has size at most $(k-1)$. Following the analysis of the greedy algorithm in [17], the total number of bins used in packing vectors in S' is bounded by $H_{k-1} \cdot \text{OPT}$ (H_i is the ith harmonic number). Putting things together, we obtain that the number of bins used by our algorithm, A, is bounded as follows:

$$A \leq m + (d \cdot m)/k + H_{k-1} \cdot \text{OPT} \leq (1 + \epsilon \cdot d + O(\ln \epsilon^{-1})) \cdot \text{OPT}.$$

This completes the proof of Theorem 5. Substituting $\epsilon = 1/d$, we obtain the following simple corollary:

Corollary 2 *For fixed d, vector bin packing can be approximated to within $O(\ln d)$ in polynomial time.*

4 Inapproximability Results

In this section we show hardness of approximation results for the three problems we consider, vector scheduling, vector bin packing, and packing integer programs. We start with PIPs.

4.1 Packing Integer Programs (PIPs)

Randomized rounding techniques of Raghavan and Thompson [27] yield integral solutions of value $t_1 = \Omega(\mathrm{OPT}/d^{1/B})$ if $A \in [0,1]^{d\times n}$, and $t_2 = \Omega(\mathrm{OPT}/d^{1/(B+1)})$ if $A \in \{0,1\}^{d\times n}$. Srinivasan [29] improved these results to obtains solutions of value $\Omega(t_1^{B/(B-1)})$ and $\Omega(t_2^{(B+1)/B})$ respectively. We show that PIPs are hard to approximate to within a factor of $\Omega(d^{\frac{1}{B+1}-\epsilon})$ for every fixed integer B. We start with the case $A \in \{0,1\}^{d\times n}$ and then indicate how our result extends to $A \in [0,1]^{d\times n}$. Our reduction uses the result of Haståd [18] that shows that unless $NP = ZPP$ the maximum independent set problem is hard to approximate within a factor of $n^{1-\epsilon}$ for any fixed $\epsilon > 0$. Since the upper bounds are in terms of d, from here on, we will express the inapproximability factor only as a function of d.

Given a graph $G = (V,E)$ with $|V| = n$ and a positive integer B, we construct an instance of an instance of a PIP, I_G, as follows. Let A be a $d \times n$ zero-one matrix with $d = n^{(B+1)}$ such that each row corresponds to an element from $V^{(B+1)}$. Let $r_i = (v_{i_1}, \ldots, v_{i_{(B+1)}})$ denote the tuple associated with the ith row of A. We set a_{ij} to 1 if and only if the following conditions hold, otherwise we set it to 0: (a) the vertex v_j occurs in r_i, and (b) the vertices in r_i induce a *clique* in G.

We set c to $\{1\}^n$ and b to $\{B\}^d$. For any fixed integer B, the reduction can be done in polynomial time. Note that a feasible solution to I_G can be described as a set of indices $S \subseteq \{1,\ldots,n\}$.

Lemma 4.1 *Let $X \subseteq V$ be an independent set of G. Then $S = \{i \mid v_i \in X\}$ is a feasible solution to I_G of value $|S| = |X|$. Furthermore S can be packed with a height bound of 1.*

Proof. Suppose that in some dimension the height induced by S is greater than 1. Let r be the tuple associated with this dimension. Then there exist $i, j \in S$ such that $v_i, v_j \in r$ and $(v_i, v_j) \in E$. This contradicts the assumption that X is an independent set. □

Lemma 4.2 *Let S be any feasible solution to I_G and let G_S be the subgraph of G induced by the set of vertices v_i such that $i \in S$. Then $\omega(G_S) \leq B$ where $\omega(G_S)$ is the clique number of G_S.*

Proof. Suppose there is a clique of size $(B+1)$ in G_S; w.l.o.g. assume that $v_1, \ldots, v_{(B+1)}$ are the vertices of that clique. Consider the tuple $(v_1, v_2, \ldots, v_{(B+1)})$ and let i be the row of A corresponding to the above tuple. Then by our construction, $a_{ij} = 1$ for $1 \leq j \leq (B+1)$. There are $(B+1)$ vectors in S with a 1 in the same dimension i, violating the ith row constraint. This contradicts the feasibility of S. $\square$

The following is a standard Ramsey type result.

Lemma 4.3 *Let G be a graph on n vertices with $\omega(G) \leq k$. Then $\alpha(G) \geq n^{1/k}$.*

Lemmas 4.2 and 4.3 give us the following corollary:

Corollary 3 *Let S be any valid solution to I_G of value $t = |S|$. Then $\alpha(G) \geq t^{1/B}$.*

Theorem 6 *Unless $NP = ZPP$, for every fixed integer B and fixed $\epsilon_0 > 0$, PIPs with bound $b = \{B\}^d$ and $A \in \{0,1\}^{d \times n}$ are hard to approximate to within a factor of $d^{\frac{1}{B+1} - \epsilon_0}$. PIPs with $A \in [0,1]^{d \times n}$ and B rational are hard to approximate to within a factor of $d^{\frac{1}{\lfloor B \rfloor + 1} - \epsilon_0}$.*

Proof. We first look at the case of PIPs with $A \in \{0,1\}^{d \times n}$. Notice that our reduction produces only such instances. Suppose there is a polynomial time approximation algorithm $\mathcal{A}$ for PIPs with bound B that has an approximation ratio $d^{\frac{1}{B+1} - \epsilon_0}$ for some fixed $\epsilon_0 > 0$. This can be reinterpreted as a $d^{\frac{1-\epsilon}{B+1}}$-approximation where $\epsilon = \epsilon_0(B+1)$ is another constant. We will obtain an approximation algorithm $\mathcal{G}$ for the maximum independent set problem with a ratio $n^{1-\delta}$ for $\delta = \epsilon/B$. The hardness of maximum independent [18] will then imply the desired result. Given a graph G, the algorithm $\mathcal{G}$ constructs an instance I_G of a PIP as described above and gives it as input to $\mathcal{A}$. $\mathcal{G}$ returns $\max(1, t^{1/B})$ as the independent set size of G where t is the value returned by $\mathcal{A}$ on I_G. Note that by Corollary 3, $\alpha(G) \geq t^{1/B}$ which proves the correctness of the algorithm. Now we prove the approximation guarantee. We are interested only in the case when $\alpha(G) \geq n^{1-\delta}$ for otherwise a trivial independent set of size 1 gives the required approximation ratio. From Lemma 4.1 it follows that the optimal value for I_G is at least $\alpha(G)$. Since $\mathcal{A}$ provides a $d^{\frac{1-\epsilon}{B+1}}$-approximation $t \geq \alpha(G)/d^{\frac{1-\epsilon}{(B+1)}}$. In the construction of I_G $d = n^{(B+1)}$. Therefore $t \geq \alpha(G)/n^{(1-\epsilon)}$. Simple algebra verifies that $t^{1/B} \geq \alpha(G)/n^{1-\delta}$ when $\alpha(G) \geq n^{1-\delta}$.

Now we consider the case of PIPs with $A \in [0,1]^{d \times n}$. Let B be some real number. For a given B we can create an instance of a PIP as before with $B' = \lfloor B \rfloor$. The only difference is that we set $b = B^d$. Since all entries of A are integral, effectively the bound is B'. Therefore it is hard to approximate to within a factor of $d^{(1-\epsilon)/(B'+1)} = d^{(1-\epsilon)/(\lfloor B \rfloor + 1)}$. Since $(\lfloor B + 1/d \rfloor + 1) = B + 1$, $d^{(1-\epsilon)/(\lfloor B \rfloor + 1)} = \Theta(d^{(1-\epsilon)/B})$. $\square$

Discussion: An interesting aspect of our reduction above is that the hardness results hold even when the optimal algorithm is restricted to a height bound of 1 while allowing a height bound of B for the approximation algorithm. Let an (α, β)-bicriteria approximation be one that satisfies the relaxed constraint matrix $Ax \leq \alpha b$ and gets a solution of value at least OPT/β, here OPT satisfies $Ax \leq b$. Then we have the following corollary:

Corollary 4 *Unless $NP = ZPP$, for every fixed integer B and fixed $\epsilon > 0$, it is hard to obtain a $(B, d^{\frac{1}{B+1}-\epsilon})$ bicriteria approximation for PIPs.*

For a given B, we use $d = n^{B+1}$, and a hardness of $d^{\frac{1}{B+1}-\epsilon}$ is essentially the hardness of $n^{1-\epsilon}$ for independent set. This raises two related questions. First, should d be larger than n to obtain the inapproximability results? Second, should the approximability (and inapproximability) results be parameterized in terms of n instead of d? These questions are important to understand the complexity of PIPs as d varies from $O(1)$ to $\text{poly}(n)$. We observe that the hardness result holds as long as d is $\Omega(n^{\epsilon})$ for some fixed $\epsilon > 0$. To see this, observe that in our reduction, we can always add $\text{poly}(n)$ dummy columns (vectors) that are either useless (their c_j value is 0) or cannot be packed (add a dummy dimension where only B of the dummy vectors can be packed). Thus we can ensure that $n \geq \text{poly}(d)$ without changing the essence of the reduction. We have a PTAS when $d = O(1)$ and a hardness result of $d^{1/(B+1)}$ when $d = \text{poly}(n)$. An interesting question is to resolve the complexity of the problem when $d = \text{polylog(n)}$.

As remarked earlier, Srinivasan [29] improves the results obtained using randomized rounding to obtain solutions of value $\Omega(t_2^{(B+1)/B})$ where $t_2 = \Omega(y^*/d^{1/(B+1)})$ for $A \in \{0,1\}^{d \times n}$. In the above y^* is the optimal fractional solution to the PIP. It might appear that this contradicts our hardness result but observe that for the instances we create in our reduction $y^*/d^{1/(B+1)} \leq 1$. For such instances Srinivasan's bounds do not yield an improvement over randomized rounding.

4.2 Vector Scheduling

We now extend the ideas used in the hardness result for PIPs to show the following hardness result for the vector scheduling problem.

Theorem 7 *Unless $NP = ZPP$, for every constant $\gamma > 1$, there is no polynomial time algorithm that approximates the schedule height in the vector scheduling problem to within a factor of γ.*

Our result here uses the hardness of graph coloring; Feige and Kilian [6] building on the work of Håstad [18] show that graph coloring is $n^{1-\epsilon}$-hard unless NP=ZPP. Our reduction is motivated by the fact that graph coloring corresponds to covering a graph by independent sets. We start with the following simple lemma that is easily derived from Lemma 4.3.

Lemma 4.4 *Let G be a graph on n vertices with $\omega(G) \leq k$. Then $\chi(G) \leq O(n^{1-1/k} \ln n)$.*

Let $B = \lceil \gamma \rceil$; we will show that it is hard to obtain a B-approximation using a reduction from chromatic number. Given graph G we construct an instance I of the VS problem as follows. We construct n vectors of n^{B+1} dimensions as in the proof of Theorem 6. We set m the number of machines to be $n^{\frac{1}{2B}}$.

Lemma 4.5 *If $\chi(G) \leq m$ then the optimal schedule height for I is 1.*

Proof. Let $V_1, \ldots, V_{\chi(G)}$ be the color classes. Each color class is an independent set and by Lemma 4.1 the corresponding vectors can be packed on one machine with height at most 1. Since $\chi(G) \leq m$ the vectors corresponding to each color class can be packed in a separate machine. □

Lemma 4.6 *If the schedule height for I is bounded by B then $\chi(G) \leq \beta n^{1-1/2B} \ln n$ for some fixed constant β.*

Proof. Let $V_1, V_2, \ldots, V_m$ be the partition of vertices of G induced by the assignment of the vectors to the machines. Let G_i be the subgraph of G induced by the vertex set V_i. From Lemma 4.2 we have $\omega(G_i) \leq B$. Using Lemma 4.4 we obtain that $\chi(G_i) \leq \beta n^{1-1/B} \ln n$ for $1 \leq i \leq m$. Therefore it follows that $\chi(G) \leq \sum_i \chi(G_i) \leq m \cdot \beta n^{1-1/B} \ln n \leq \beta n^{1-1/2B} \ln n$. □

Proof of Theorem 7. Feige and Kilian [6] showed that unless $ZPP = NP$ for every $\epsilon > 0$ there is no polynomial time algorithm to approximate the chromatic number to within a factor of $n^{1-\epsilon}$. Suppose there is a B-approximation for the VS problem. Lemmas 4.5 and 4.6 establish that if $\chi(G) \leq n^{1/2B}$ then we can infer by running the B-approximation algorithm for the VS problem that $\chi(G) \leq \beta n^{1-1/2B} \ln n$. This implies a $\beta n^{1-1/2B} \ln n$-approximation to the chromatic number. From the result of [6] it follows that this is not possible unless $NP = ZPP$. □

4.3 Vector Bin Packing

As mentioned before VBP is APX-hard even for $d = 2$. We show a simple $d^{\frac{1}{2}-\epsilon}$ hardness for VBP when d is arbitrary. The reduction is similar to that in the previous subsection and uses hardness of graph coloring. Given a graph G with n vertices and m edges we create an instance I_G of VBP with n vectors, each of m dimensions. For a vector v_i corresponding to a vertex i of G, the jth coordinate is 1 if i is incident on the jth edge, otherwise it is 0. If the vectors are required to be packed into bins of height 1 it is easily seen that the number of bins required corresponds exactly to a coloring of G. Thus the hardness of chromatic number applies directly to VBP.

Theorem 8 *Unless $NP = ZPP$, VBP is hard to approximate to within a $d^{\frac{1}{2}-\epsilon}$ factor for every fixed $\epsilon > 0$.*

5 Conclusions

We studied multi-dimensional generalizations of multiprocessor scheduling, bin packing and the knapsack problem and obtained a variety of new algorithmic as well as inapproximability results for them. While our work gives news insights into the approximability of these problems, several questions remain open. In particular, large gaps remain between the upper and lower bounds for both vector scheduling and vector bin packing when the number of dimensions is arbitrary. For packing integer programs, our hardness result is essentially tight, but it applies only to fixed values of B. The quality of approximation improves dramatically when B is allowed to grow as a logarithmic function of d, in particular a constant factor approximation is achievable when $B = \Omega(\log d/\log\log d)$. It will be interesting to see if our techniques can be extended to obtain hardness result when B is allowed to be a function of d.

Acknowledgments

We are grateful to Aravind Srinivasan for guiding us through the literature on packing integer programs and his encouragement at the early stages of this research. We thank Cliff Stein for his comments on this work in its form as a chapter in the dissertation of the first author. We also thank the anonymous referees for many useful comments on an earlier version of the paper.

References

[1] N. Alon, Y. Azar, J. Csirik, L. Epstein, S. V. Sevastianov, A. P. A. Vestjens, and G. J. Woeginger. On-line and off-line approximation algorithms for vector covering problems. *Algorithmica*, 21(1):104–18, 1998.

[2] S. F. Assman. *Problems in Discrete Applied Mathematics.* PhD thesis, Mathematics Dept, MIT, 1983.

[3] C. Chekuri and S. Khanna. On multi-dimensional packing problems. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, 185–194, 1999.

[4] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

[5] D. J. DeWitt and J. Gray. Parallel database systems: The future of high performance database systems. *Communications of the ACM*, 35(6):85–98, June 1992.

[6] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *J. of Computer and System Sciences*, 57(2):187–199, 1998. Preliminary version in *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity*, pages 278–287, 1996.

[7] A. M. Frieze and M. R. B. Clarke. Approximation algorithms for the m-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses. *European Journal of Operational Research*, 15(1):100–9, 1984.

[8] M. R. Garey and R. L. Graham. Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, 4(2):187–200, June 1975.

[9] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C. Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory Ser. A*, 21:257–298, 1976.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, 1979.

[11] Minos N. Garofalakis and Yannis E. Ioannidis. Scheduling issues in multimedia query optimization. *ACM Computing Surveys*, 27(4):590–92, December 1995.

[12] Minos N. Garofalakis and Yannis E. Ioannidis. Multi-dimensional resource scheduling for parallel queries. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 365–76, June 1996.

[13] Minos N. Garofalakis and Yannis E. Ioannidis. Parallel query scheduling and optimization with time-and space-shared resources. In *Proceedings of the 23rd VLDB Conference*, pages 296–305, 1997.

[14] Minos N. Garofalakis, Banu Özden, and Avi Silberschatz. Resource scheduling in enhanced pay-per-view continuous media databases. In *Proceedings of the 23rd VLDB Conference*, pages 516–525, 1997.

[15] R. L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Tech. J.*, 45:1563–81, 1966.

[16] R. L. Graham. Bounds on multiprocessor timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.

[17] M. M. Halldórsson. Approximating k-set cover and complementary graph coloring. In *Proceedings of Fifth IPCO Conference on Integer Programming and Combinatorial Optimization*, LNCS 1084, pages 118–131. Springer Verlag, 1996.

[18] J. Håstad. Clique is hard to approximate to within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999. Preliminary version in *Proceedings of the 37th Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[19] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

[20] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–8, 1975.

[21] N. Karmarkar and R. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Symposium on Foundations of Computer Science*, pages 312–320, 1982.

[22] R. M. Karp, M. Luby, and A. Marchetti-Spaccamela. A probabilistic analysis of multi-dimensional bin packing problems. In *Proceedings of the Annual ACM Symposium on the Theory of Computing*, pages 289–298, 1984.

[23] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *Handbooks in OR & MS*, volume 4, chapter Sequencing and Scheduling: Algorithms and Complexity, pages 445–522. Elsevier Science Publishers, 1993.

[24] L. Lovász. On the ratio of the optimal integral and fractional covers. *Discrete Math.*, 13:383–390, 1975.

[25] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press (1995).

[26] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[27] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[28] Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience series in discrete mathematics. Wiley, 1986.

[29] Aravind Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM J. on Computing*, 29(2):648–670, 1999. Preliminary version in *Proceedings of the 27th ACM Symposium on the Theory of Computing*, pages 268–276, 1995.

[30] G. Woeginger. A polynomial time approximation scheme for maximizing the minimum completion time. *Operations Research Letters*, 20:149–154, 1997.

[31] G. Woeginger. There is no asymptotic PTAS for two-dimensional vector packing. *Information Processing Letters*, 64:293–97, 1997.