On Multi-dimensional Packing Problems

Chandra Chekuri* Sanjeev Khanna†

Abstract

We study the approximability of multi-dimensional generalizations of the classical problems of 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.

We obtain a variety of new algorithmic as well as inapproximability results for these problems. For vector scheduling, we give a PTAS when d is a fixed constant, and an $O(\min\{\log dm, \log^2 d\})$-approximation in general. We also show that unless NP = ZPP, no polynomial time algorithm can give a γ-approximation for any constant γ. For vector bin packing, we obtain a $(1+\epsilon \cdot d + O(\ln \epsilon^{-1}))$-approximation for any fixed $\epsilon > 0$, thus significantly improving upon the earlier bound of $(d+\epsilon)$ [6, 11]. This result also implies an $O(\ln d)$-approximation for the case when d is a fixed constant. On the other hand, we show that vector bin packing is APX-hard even when $d = 2$.

A core problem that directly relates to both vector scheduling and vector bin packing is the approximability of packing a maximum number of vectors in a single bin of unit height. This problem is a special case of the packing integer programs (PIPs) where we are given a matrix $A \in \{0,1\}^{d \times n}$ ($A \in [0,1]^{d \times n}$), a vector $b \in [1, \infty)^d$ and $c \in [0,1]^n$ (with $\max_j c_j = 1$). We show that unless NP = ZPP, PIPs are hard to approximate to within a factor of $\Omega(d^{\frac{1}{B+1}-\epsilon})$ ($\Omega(d^{\frac{1}{\lfloor B \rfloor + 1}-\epsilon})$) for any fixed $\epsilon > 0$, where $B = \min_i b_i$ is some fixed integer (rational). This essentially matches the best known approximation ratio for PIPs, obtained via randomized rounding [29, 30]. 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 $Ax \leq B^d$.

*Department of Computing Principles Research, 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 a student 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.

†Department of Fundamental Mathematics Research, Bell Labs, 700 Mountain Avenue, Murray Hill, NJ 07974. E-mail: sanjeev@research.bell-labs.com. URL: http://cm.bell-labs.com/who/sanjeev.

1 Introduction

Multi-processor scheduling, bin packing, and the knapsack problem are three 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 in fixed dimension, and an approximation algorithm for the vector bin-packing problem that substantially 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 the multi-dimensional knapsack problem that is equivalent to packing integer programs (PIPs) [29, 30]. PIPs are an important class of integer programs that capture several NP-hard combinatorial optimization problems in graphs and hypergraphs including maximum independent set, disjoint paths and hypergraph matchings. The only general technique known for approximating PIPs is to use randomized rounding on the natural LP relaxation [29, 30]. We resolve here the question of whether there exists a better approximation guarantee for PIPs. Unless NP=ZPP, we show that no polynomial time algorithm can achieve a better approximation ratio.

Besides having 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 [14, 15, 16] in multi-dimensional resource scheduling problems in parallel query optimization. A favored architecture for parallel databases is the so called shared-nothing environment [7] where the parallel system consists of a set of independent processing units each of which has a set of time-shareable 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. For large task systems that are typically encountered in database applications, however, ignoring the multi-dimensionality could lead to bad performance. The work in [13, 14, 15, 16] 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 [17, 25], the authors in [13] had to settle for a naive $(d+1)$ approximation for the above problem. In contrast, our work here provides a PTAS for every fixed d and an $O(\log^2 d)$-approximation when d is arbitrary. A similar situation existed for the dual vector bin packing problem where the best known approximation ratio is a $(d+\epsilon)$-approximation. In this paper, we improve this to 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 $\max_{1\leq i\leq m} \|\bar{A}_i\|_\infty$ 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 $\|\bar{A}_i\|_\infty \leq 1$ for $1 \leq j \leq m$. The objective is to minimize m, the size of the partition.*

The following definition of PIPs is from [30]. In the literature this problem is also referred to as the d-dimensional 0-1 knapsack problem [9].

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 $c^T \cdot x$ subject to $x \in \{0,1\}^n$ and $Ax \leq b$. 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 [30]). We are interested in PIPs where $b_i = B$ for $1 \leq i \leq d$. When $A \in \{0,1\}^{d\times n}$ this problem is known as the *simple B-matching in hypergraphs* [26]: 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 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 [15] 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), thus generalizing the result of Hochbaum and Shmoys [21] for multiprocessor scheduling. In addition we obtain a simpler $O(\log d)$ approximation algorithm that is better than $(d+1)$ for all $d \geq 2$. When d is large we give an $O(\log^2 d)$ approximation that uses as a subroutine, known approximation algorithms for PIPs. We also give a very simple $O(\log dm)$-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 $\epsilon > 0$ [6] and $(d + 7/10)$ [11]; the latter result holds even in an online setting. All the ratios mentioned are asymptotic, that is there is an additive term of d. Karp et al. [24] 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 for a fixed d, we can approximate the problem to within a ratio of $O(\ln d)$. We also note that if the input vectors are drawn from the set $[0,1]^d$, the approximation ratio can be improved to $O(\sqrt{d})$. This latter result is essentially tight since via a reduction from graph coloring, we show a $d^{1/2-\epsilon}$-hardness for this problem, for any fixed $\epsilon > 0$. Moreover, we show that even for $d = 2$ the problem is APX-hard; an interesting departure from classical bin packing problem ($d = 1$) which exhibits an asymptotic FPTAS. Our hardness reduction here also gives us an APX-hardness result for the so-called *vector covering problem* (also known as the "dual vector packing" problem). Only an NP-hardness result was known for this problem prior to our work.

Packing Integer Programs: For fixed d there is a PTAS for PIPs [9]. For large d randomized rounding technique of Raghavan and Thompson [29] yields integral solutions of value $t_1 = \Omega(\mathrm{OPT}/d^{1/B})$ and $t_2 = \Omega(\mathrm{OPT}/d^{1/(B+1)})$ respectively, if $A \in [0,1]^{d\times n}$ and $A \in \{0,1\}^{d\times n}$. Srinivasan [30] improved these results to obtains 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 $B = 1$ earlier. 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 $Ax \leq B^d$. We use Haståd's recent result [20] on the inapproximability of the maximum independent set 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 present our hardness of approximation results for PIPs, vector scheduling, and vector bin packing and covering.

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 $\bar{A}$. The sum of the components of a vector plays a role in the algorithms. Since the vectors are all positive this is simply the ℓ_1 norm of the vector. Throughout this section, we assume w.l.o.g. that the optimal schedule value is 1.

2.1 A PTAS for fixed d

Hochbaum and Shmoys [21] gave a PTAS for the multiprocessor 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 [21] 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 jobs into large or small depending on whether their 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 discretized into $O(\log 1/\epsilon)$ classes and dynamic programming is used to pack all the 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 view of 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 vectors into large or small depending on their ℓ_∞ norm but the scheme of [21] 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 us define δ to 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'.*

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 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 in to 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 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 bin configuration $(a_1, \ldots, a_d)$ if the the height of the packing is 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 reforms 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 $q = \lceil \frac{2}{\epsilon} \log \delta^{-1} \rceil$ 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'.*

Vectors in L' can be classified into one of $s = (1 + \lceil \frac{2}{\epsilon} \log \delta^{-1} \rceil)^d$ 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 $\sum k_i \le d/\delta$. 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 earlier $|C_k| \le (d/\delta)^s$.

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 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 with each bin assigns a capacity configuration from M. For $1 \le i \le m$, the algorithm computes all possible subsets of vectors from L' (tuples) that can be packed 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.

$$\sum_j x_{ij} = 1 \qquad 1 \leq i \leq |S| \tag{2.1}$$

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

$$\sum_i p_i^k \cdot x_{ij} \leq b_j^k \qquad 1 \leq j \leq m, 1 \leq k \leq d. \tag{2.2}$$

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.

$$x_{ij} \geq 0 \tag{2.3}$$

PROPOSITION 2.1. *Any* basic feasible solution *to the LP defined by Equations 2.1, 2.2, and 2.3 has at most $d \cdot m$ vectors that are assigned fractionally to more than one machine.*

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 assign the vectors in S' in a round robin fashion to the machines that ensures that no machine gets more than d vectors each from S'. However since $\|p_j\|_\infty \leq \delta = \epsilon/d$ for every $p_j \in S'$, the above step does not violate the height by more than ϵ in any dimension.

Putting it Together: We are now ready to prove our main theorem.

THEOREM 2.1. *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 $s = O((\frac{\log(d/\epsilon)}{\epsilon})^d)$.*

Proof Sketch. 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 bound follows using Lemma 2.3. □

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 ($O(\log^2 d)$) while the second algorithm is randomized and achieves an approximation ratio that is a function of both d and m ($O(\log dm)$). We once again assume that the optimal schedule height is 1.

2.2.1 An $O(\log^2 d)$ Approximation: We defer the technical details and sketch here the main ideas underlying our approximation. Define the *volume* of a vector to be sum of its coordinates, and the volume of a set of vectors A, denoted by $\mathcal{V}(A)$, to be the sum of the volumes of the vectors in A. Consider an optimal schedule for a given instance. By simple averaging arguments, some machine k in that schedule satisfies the condition

$$\mathcal{V}(J(k)) \geq \mathcal{V}(J)/m,$$

where $J(k)$ is the set of vectors assigned to machine k and J is the set of all the input vectors. We use approximation algorithms for PIPs to find a set of jobs of total volume $\mathcal{V}(J)/m$ that can be packed on a machine with height close to 1. We then remove these jobs and find another set of jobs to pack in the second machine, and so on (that is if jobs are left). A stage ends when we have used all the m machines. Standard arguments allow us to argue that the volume of the jobs left after a stage is at most a constant factor of the initial volume. After $\Theta(\log d)$ stages the volume of the jobs not scheduled is reduced to less than $\mathcal{V}(J)/d$. Then we use a naive list scheduling algorithm on the remaining jobs; th edetails are defered.

For arbitrary d, the PIP formulation essentially yields an $O(\log d)$-approximation to the volume packing problem [28, 29, 30]. On the other hand, when d is fixed, the volume packing problem has a PTAS [9]. These observations can be used to obtain the following two corollaries.

COROLLARY 2.1. *There is an $O(\log^2 d)$ approximation algorithm for the VS problem.*

COROLLARY 2.2. *There is an $O(\log d)$ approximation algorithm for the VS problem that runs in time polynomial in n^d. Thus for fixed d, we obtain an $O(\log d)$-approximation in polynomial time.*

2.2.2 An $O(\log dm)$ Approximation: The approximation in Corollary 2.1 is good when d is small compared to m. However when d is large we can obtain a $O(\log dm)$ approximation by a simple randomized algorithm that assigns each vector independently to a machine chosen uniformly at random from the set of m machines. Using Chernoff bounds, it is easy to show the following:

THEOREM 2.2. *Random gives an $O(\log dm)$ approximation with high probability.*

3 Algorithms for Vector Bin Packing

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

THEOREM 3.1. *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 [6]. 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 $m \leq \text{OPT}$.

$$\sum_j x_{ij} = 1 \qquad 1 \leq i \leq n$$

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

$$x_{ij} \geq 0 \qquad 1 \leq i \leq n\ ,\ 1 \leq j \leq m$$

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 [19], 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 3.1. Substituting $\epsilon = 1/d$, we obtain the following simple corollary:

COROLLARY 3.1. *For any arbitrary but fixed d, vector bin packing can be approximated to within $O(\ln d)$ in polynomial time.*

Using a simple argument, the result of the previous theorem can be strengthened to an $O(\sqrt{d})$-approximation when the vectors are drawn from the space $\{0,1\}^n$; the proof is omitted.

THEOREM 3.2. *If each vector $p_i \in \{0,1\}^n$, then we can obtain a $(2\sqrt{d})$-approximation for vector bin packing.*

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 [29] yield integral solutions of value $t_1 = \Omega(\text{OPT}/d^{1/B})$ and $t_2 = \Omega(\text{OPT}/d^{1/(B+1)})$ respectively, if $A \in [0,1]^{d \times n}$ and $A \in \{0,1\}^{d \times n}$. Srinivasan [30] 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 recent result of Haståd [20] that shows that independent set is hard to approximate within a factor of $n^{1-\epsilon}$ for any fixed $\epsilon > 0$, unless $NP = ZPP$. 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 and a positive integer B, we construct an instance of an instance of a PIP I_G as follows. Create a $d \times n$ zero-one matrix A 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 $\omega(G_S) \le B$ 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 $1 \le j \le (B+1)$. 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. □

The following is a simple Ramsey type result; we prove it here for the sake of completeness.

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

Proof. By induction on k. Base case with $k = 1$ is trivial. Assume hypothesis is true for integers up to $k-1$. Consider a graph with $\omega(G) = k$. If the degree of every vertex in G is less than $n^{(k-1)/k}$, then any maximal independent set has size at least $n^{1/k}$. Otherwise, consider a vertex v in G that has degree at least $n^{(k-1)/k}$. Let G' be the subgraph of G induced by the neighbors of v. Since $\omega(G') \le k-1$, by the induction hypothesis, $\alpha(G') \ge (n^{(k-1)/k})^{1/(k-1)} \ge n^{1/k}$. □

Lemmas 4.2 and 4.3 give us the following corollary:

COROLLARY 4.1. *Let S be any valid solution to I_G of value $t = |S|$. Then $\alpha(G) \ge t^{1/B}$.*

THEOREM 4.1. *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 [20] 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 4.1, $\alpha(G) \ge t^{1/B}$ which proves the correctness of the algorithm. Now we prove the approximation guarantee. We are interested only in the case when $\alpha(G) \ge n^{1-\delta}$ 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 $t \ge \alpha(G)/d^{\frac{1-\epsilon}{(B+1)}}$. In the construction of I_G $d = n^{(B+1)}$. Therefore $t \ge \alpha(G)/n^{(1-\epsilon)}$. Simple algebra verifies that $t^{1/B} \ge \alpha(G)/n^{1-\delta}$ when $\alpha(G) \ge n^{1-\delta}$.

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 $d^{(1-\epsilon)/(B'+1)} = d^{(1-\epsilon)/(\lfloor B \rfloor + 1)}$. If $(\lfloor B + 1/d \rfloor + 1) = B + 1$ then $d^{(1-\epsilon)/(\lfloor B \rfloor + 1)} = \Theta(d^{(1-\epsilon)/B})$. □

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 $Ax \le \alpha b$ and gets a solution of value at least OPT$/\beta$, here OPT satisfies $Ax \le b$. Then we have the following corollary:

COROLLARY 4.2. *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 [30] 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 4.2. *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 [8] building on the work of Haståd [20] 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.

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

Proof. From Lemma 4.3 $\alpha(G) \geq n^{1/k}$. Let G' be the graph obtained by removing a largest independent set from G. It is easy to see that $\omega(G') \leq k$. Thus we can apply Lemma 4.3 again to G' to remove another large independent set. We can repeat this process until we are left with a single vertex and standard arguments show that the process terminates after $O(n^{1-1/k}\log n)$ steps. Thus we can partition $V(G)$ into $O(n^{1-1/k}\log n)$ independent sets and the lemma follows. □

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 4.1. 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 $\chi(G) \leq \beta n^{1-1/2B}\log n$ 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 $\chi(G_i) \leq \beta n^{1-1/B}\log n$ for $1 \leq i \leq m$. Therefore it follows that $\chi(G) \leq \sum_i \chi(G_i) \leq m \cdot \beta n^{1-1/B}\log n \leq \beta n^{1-1/2B}\log n$. □

Proof of Theorem 4.2. Feige and Kilian [8] 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 $\chi(G) \leq \beta n^{1-1/2B}\log n$. This implies a $\beta n^{1-1/2B}\log n$ approximation to the chromatic number. From the result of [8] it follows that this is not possible unless $NP = ZPP$. □

4.3 Vector Bin Packing (and Vector Covering)

In this section we prove that the vector bin packing problem is APX hard even for $d = 2$. In contrast the standard bin packing with $d = 1$ has an asymptotic PTAS [6, 23]. Thus our result shows that bin packing with $d > 1$ has no PTAS and in particular no asymptotic PTAS. Using a similar reduction we obtain APX hardness for the vector covering problem [1] as well. We use a reduction from the optimization version of bounded 3-Dimensional matching (3DM) problem [12]. We define the problem formally below.

DEFINITION 4.1. (3-BOUNDED 3DM (3DM-3)) *Given a set $T \subseteq X \times Y \times Z$. A matching in T is a subset $M \subseteq T$ such that no elements in M agree in any coordinate. The goal is to find a matching in T of largest cardinality. A* 3*-bounded instance is one in which the number of occurrences of any element of $X \cup Y \cup Z$ in T is at most* 3.

Kann [22] showed that the above problem is Max-SNP Complete (hence also APX-Complete). Our reduction builds on the ideas used in the NP-Completeness reduction that reduces the 3DM problem to the 4-Partition problem [12].

Let $q = \max\{|X|, |Y|, |Z|\}$, $u = |X \cup Y \cup Z|$ and $t = |T|$. We create a total of $(u + t)$ 2-dimensional vectors. We have one vector each for the elements of T and one for each of the elements of $X \cup Y \cup Z$. Let x_i, y_j, z_k, w_l denote the corresponding vectors for X, Y, Z, and T respectively. The first coordinates are assigned as follows.

$$\begin{array}{rcll} x_i^1 & = & q^4 + i & 1 \le i \le |X| \\ y_j^1 & = & q^4 + jq & 1 \le j \le |Y| \\ z_k^1 & = & q^4 + kq^2 & 1 \le k \le |Z| \end{array}$$

The first coordinate of an element $l = (i, j, k)$ in T is assigned as $w_l^1 = q^4 - kq^2 - jq - i$.

The second dimension of each of the vectors is obtained simply by subtracting the first dimension from $2q^4$. We will assume without loss of generality that $q \ge 4$ in which case all coordinates are positive. Our goal is to pack these vectors in two dimensional bins with capacity $4q^4$. Our construction has the following properties.

PROPOSITION 4.1. *Any* three *vectors in the instance can be packed in a bin.*

An element of T is referred to as a *hyper-edge*. The elements of a hyper-edge e are the elements of $X \cup Y \cup Z$ that form the tuple e.

PROPOSITION 4.2. Four *vectors can be packed in a bin if and only if they correspond to a hyper-edge and its elements.*

THEOREM 4.3. *Vector bin packing in* 2 *dimensions is APX-Complete.*

Proof. VBP in 2 dimensions is in APX by Theorem 3.1. We will show APX hardness via an L-reduction [27] from 3-bounded 3DM to VBP with $d = 2$. Let $\mathcal{I}$ be an instances of 3DM-3. We map $\mathcal{I}$ to an instance $\mathcal{I}'$ of VBP as described in our reduction above. Let m^* be the size of the largest matching in $\mathcal{I}$ and let b^* be the smallest number of bins needed for $\mathcal{I}'$. By Propositions 4.2 and 4.1 it is easily verified that $b^* = \lceil (t + u - m^*)/3 \rceil$. We claim that there exists a constant $\alpha > 0$ such that $b^* \le \alpha m^*$ for any instance of 3DM-3. This follows from the fact that $u = O(t)$ (due to 3-boundedness) and $t = O(m^*)$ (due to membership in Max-SNP). Now to map a solution for $\mathcal{I}'$ to a solution for $\mathcal{I}$, we define the matching to be the hyper-edges corresponding to the vectors in T that are packed in bins with 4 vectors each. Proposition 4.2 guarantees the correctness of the solution. We assume without loss of generality that all bins except one have at least three vectors each. Thus if m is the number of bins with 4 vectors packed, then b, the total number of bins, is $\lceil (t + u - m)/3 \rceil$. Thus we obtain that

$$\begin{array}{rcl} b - b^* & = & \lceil (t + u - m)/3 \rceil - \lceil (t + u - m^*)/3 \rceil \\ & \ge & (t + u - m)/3 - (t + u - m^*)/3 + 1 \\ & \ge & (m^* - m)/3 + 1 \end{array}$$

In other words, there exists a constant β such that for all instances $|m^* - m| \le \beta |b^* - b|$. It follows that 3DM-3 L-reduces to VBP. □

Vector Covering: A similar reduction as above shows hardness of the *vector covering problem* defined below.

DEFINITION 4.2. (VECTOR COVERING (VC)) *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 $\bar{A}_j^i \ge 1$ for $1 \le i \le d$ and $1 \le j \le m$. The objective is to maximize m, the size of the partition.*

The vector covering problem is also referred to as the dual bin packing problem and the one-dimensional version was first investigated in the thesis of Assman [2]. Both on-line and off-line versions have been studied in various papers [3, 5, 4, 1, 31] mostly for the one-dimensional case. We concentrate on the off-line approximability of the problem. For $d = 1$, a PTAS was obtained by Woeginger [31] that improved upon the constant factor algorithms of Assman et al. [3]. Alon et al. [1] gave a $\min\{d, 2\ln d/(1 + o(1))\}$ approximation algorithm for arbitrary d. However no hardness of approximation results were known thus far. Building on the ideas used in the preceding reduction, we can show the following:

THEOREM 4.4. *The Vector covering problem is APX-Complete for $d = 2$.*

References

[1] N. Alon, J. Csirik, S. V. Sevastianov, A. P. A. Vestjens, and G. J. Woeginger. On-line and off-line approximation algorithms for vector covering problems. *Algorithmica*, 1998.

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

[3] S. F. Assman, D. S. Johnson, D. J. Kleitman, and J. Y. T. Leung. On a dual version of the one-dimensional bin packing problem. *Journal of Algorithms*, 5:502–525, 1984.

[4] J. Csirik, J. B. G. Frenk, G. Galambos, and A. H. G. Rinnooy Kan. Probabilistic analysis of algorithms for dual bin packing problems. *Journal of Algorithms*, 12(2):189–203, 1991.

[5] J. Csirik and V. Totik. On-line algorithms for a dual version of bin packing. *Discrete Applied Mathematics*, 21:163–67, 1988.

[6] 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.

[7] 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.

[8] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity*, pages 278–287, 1996.

[9] 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.

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

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

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

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

[19] 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.

[20] J. Håstad. Clique is hard to approximate to within $n^{1-\epsilon}$. In *Proceedings of the 37th Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[21] 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.

[22] V. Kann. Maximum bounded 3-dimensional matching is max snp-complete. *Information Processing Letters*, 37:27–35, 1991.

[23] 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.

[24] 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.

[25] 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.

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

[27] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–40, 1991.

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

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

[30] Aravind Srinivasan. Improved approximations of packing and covering problems. In *Proceedings of the 27th ACM Symposium on the Theory of Computing*, pages 268–276, 1995.

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