On-line and Off-line Approximation Algorithms for Vector Covering Problems

NOGA ALON * YOSSI AZAR † JÁNOS CSIRIK ‡ LEAH EPSTEIN §

SERGEY V. SEVASTIANOV ¶ ARJEN P.A. VESTJENS ‖

GERHARD J. WOEGINGER **

Abstract

This paper deals with *vector covering problems* in d-dimensional space. The input to a vector covering problem consists of a set X of d-dimensional vectors in $[0,1]^d$. The goal is to partition X into a maximum number of parts, subject to the constraint that in every part the sum of all vectors is at least one in every coordinate. This problem is known to be NP-complete, and we are mainly interested in its on-line and off-line approximability.

For the on-line version, we construct approximation algorithms with worst case guarantee arbitrarily close to $1/(2d)$ in $d \geq 2$ dimensions. This result contradicts a statement of Csirik and Frenk (1990) in [5] where it is claimed that for $d \geq 2$, no on-line algorithm can have a worst case ratio better than zero. Moreover, we prove that for $d \geq 2$, no on-line algorithm can have worst case ratio better than $2/(2d+1)$. For the off-line version, we derive polynomial time approximation algorithms with worst case guarantee $\Theta(1/\log d)$. For $d = 2$, we present a very fast and very simple off-line approximation algorithm that has worst case ratio $1/2$. Moreover, we show that a method from the area of compact vector summation can be used to construct off-line approximation algorithms with worst case ratio $1/d$ for every $d \geq 2$.

Keywords. Approximation algorithm, worst case ratio, competitive analysis, on-line algorithm, packing problem, covering problem.

*`noga@math.tau.ac.il`. Department of Mathematics, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. Supported by a USA-Israeli BSF grant.

†`azar@math.tau.ac.il`. Department of Computer Science, Tel-Aviv University, Tel Aviv, Israel. Supported by Alon Fellowship and by the Israel Science Foundation, administered by the Israel Academy of Sciences.

‡`csirik@inf.u-szeged.hu`. Department of Computer Science, University of Szeged, Aradi vértanúk tere 1, H–6720 Szeged, Hungary. Supported by Project 20u2 of the Austro-Hungarian Action Fund.

§`lea@math.tau.ac.il`. Department of Computer Science, Tel-Aviv University, Tel Aviv, Israel.

¶`seva@math.nsk.ru`. Institute of Mathematics, Siberian Branch of the Russian Academy of Sciences, Universitetskii pr. 4, 630090, Novosibirsk-90, Russia. Supported by the DIMANET/PECO Program of the European Union.

‖`arjenv@win.tue.nl`. Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands. Research partially supported by the HCM-project CHRX-CT93-0087 of the European Union.

**`gwoegi@opt.math.tu-graz.ac.at`. Institut für Mathematik B, TU Graz, Steyrergasse 30, A-8010 Graz, Austria. Research partially supported by a research fellowship of the Euler Institute for Discrete Mathematics and its Applications, by the HCM-project CHRX-CT93-0087 of the European Union, by the Project P10903-MAT of the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung, by Project 20u2 of the Austro-Hungarian Action Fund, and by the Spezialforschungsbereich F003 “Optimierung und Kontrolle”, Projektbereich Diskrete Optimierung.

1 Introduction

Problem statement. For a set $S \subseteq [0,1]^d$ and a coordinate i, $1 \leq i \leq d$, we denote by $D_i(S)$ the sum of the i-th components of the elements in S. The set S is called a *unit cover* if $D_i(S) \geq 1$ holds for all $1 \leq i \leq d$; in other words, the elements of a unit cover can be used to cover simultaneously all sides of the d-dimensional unit-bin. The d-dimensional *vector covering problem* is that of partitioning a given list X of vectors in $[0,1]^d$ into the maximum number of unit covers, i.e. covering the maximum number of unit-bins with the elements of X.

This vector covering problem models a variety of situations encountered in business and in industry, from packing peach slices into tin cans so that each tin can contains at least its advertised net weight, to such complex problems as breaking up monopolies into smaller companies, each of which is large enough to be viable.

Approximation algorithms. The vector covering problem is easily seen to be NP-complete. This suggests to look for fast approximation algorithms that come close to the optimum solution in polynomial time. Let $\text{OPT}(X)$ denote the number of unit covers in an optimum partition of list X. For an approximation algorithm A, let $A(X)$ denote the number of unit covers that A produces on input list X. Define for every $k \geq 1$

$$R_A(k) \ := \ \min\left\{ \frac{A(X)}{\text{OPT}(X)} \ \mid \ \text{OPT}(X) = k \right\}. \tag{1}$$

The *asymptotic worst case ratio* R_A (also called *worst case performance*, *worst case guarantee*, or just *worst case ratio*) of an approximation algorithm A, is defined by

$$R_A \ = \ \liminf_{k\to\infty} \ R_A(k). \tag{2}$$

Clearly, $R_A(k) \leq 1$ for every $k \geq 1$ and hence $R_A \leq 1$. The asymptotic worst case ratio is the usual measure for the quality of an approximation algorithm for covering problems: the larger the ratio, the better the approximation algorithm.

Now assume an environment where the list X of vectors $x^t \in [0,1]^d$, $1 \leq t \leq n$, arrives one by one. When vector x^t arrives, it must immediately and irrevocably be assigned to its unit-bin, and the next vector x^{t+1} becomes only known *after* vector x^t has been assigned. Such an environment is called an *on-line* environment, and an approximation algorithm that is able to work in an on-line environment is called an *on-line* algorithm. In contrast to this type of algorithm are the *off-line* algorithms that only work under full knowledge of the problem data.

Known results. The *one-dimensional* version of the vector covering problem was for the first time investigated in the thesis [2] of Assmann and in the journal article by Assmann, Johnson, Kleitman and Leung [3]. There it is proved that the greedy algorithm (that simply keeps putting items into the same bin until this bin is covered and then moves on to the next bin) has a worst case guarantee of $\frac{1}{2}$. Moreover, two more sophisticated algorithms were derived with worst case ratios $\frac{2}{3}$ and $\frac{3}{4}$, respectively. Both of these sophisticated algorithms are based on presorting the items, and consequently are off-line algorithms. The greedy algorithm, however, is an on-line algorithm. Csirik and Totik [7] proved that in fact the greedy algorithm is a *best possible* on-line algorithm, since no on-line algorithm can have a worst case ratio that is strictly greater than $\frac{1}{2}$.

Csirik, Frenk, Galambos and Rinnooy Kan [6] gave a probabilistic analysis of the one-dimensional bin covering and of the two-dimensional vector covering problem. Gaizer [9] constructed an off-line approximation algorithm with worst case guarantee $\frac{1}{2}$ for dimension $d = 2$.

The article by Csirik and Frenk [5] summarizes all results on vector covering problems that were derived till 1990 (actually, there are not too many of them).

New results. In this paper, we derive five new results on vector covering problems.

(1) For every $d \geq 2$, we construct an on-line approximation algorithm with worst case ratio arbitrarily close to $1/(2d)$.

(2) For every $d \geq 2$, we show that no on-line algorithm can have a worst case ratio greater than $2/(2d+1)$.

(3) For every $d \geq 2$, we show the existence of a polynomial time off-line approximation algorithm with worst case ratio $(1+o(1))/(2\ln d)$.

(4) For $d = 2$, we present a very fast and very simple off-line approximation algorithm with worst case ratio $\frac{1}{2}$.

(5) For $d \geq 2$, we show how a method from the area of compact vector summation (see Sevastianov [13, 14]) can be used to construct off-line approximation algorithms with worst case ratio $1/d$.

Result (1) provides the first non-trivial result on on-line approximations in dimensions $d \geq 2$. It also contradicts a statement of Csirik and Frenk [5] who claim that for $d \geq 2$, no on-line algorithm can have a worst case ratio better than zero. Result (2) is a higher dimensional counterpart to the 1/2 lower bound of Csirik and Totik for $d = 1$. It also demonstrates that Result (1) is not far from being best possible. Results (3), (4) and (5) give the first off-line polynomial time approximability results for dimensions $d \geq 2$. Result (3) is based on probabilistic arguments. Result (4) uses simple ad hoc arguments to simplify the ideas of Gaizer [9]. Result (5) outperforms result (3) for several small values of d.

Other related results. The related *vector packing* problem asks for a partition of a list X of vectors into the *minimum* number of parts such that in every part the sum of all items is *at most* one in every coordinate; cf. Garey, Graham, Johnson and Yao [10]. The best known polynomial time approximation algorithms for the off-line version of d-dimensional packing have worst case guarantees $d + \varepsilon$, where $\varepsilon > 0$ is an arbitrarily small positive real (Fernandez de la Vega and Lueker [8]). The best on-line approximation algorithms have a worst case ratio of $d + \frac{7}{10}$ for $d \geq 2$ (Garey, Graham, Johnson and Yao [10]) and 1.589 for $d = 1$ (Richey [12]).

Since the vector covering problem may be considered to be a kind of inverse or dual version of the vector packing problem, it is sometimes also called "dual bin-packing" or "dual vector packing" in the literature.

Organization of the paper. Section 2 summarizes some basic notation that is used throughout the rest of the paper. Section 3 investigates an auxiliary problem that deals with the on-line partitioning of vector sets subject to a certain min-max criterion. The results derived for this auxiliary problem then constitute the main ingredients for the on-line approximation algorithm for vector covering as described in Section 4. Section 5 presents the $2/(2d+1)$ lower bound for on-line algorithms in dimensions $d \geq 2$. Section 6 gives a combinatorial lower bound for the optimum objective value. The proof of this combinatorial lower bound is probabilistic and can be translated into an efficient off-line approximation algorithm with worst case ratio $\Omega(1/\log d)$ for d-dimensional vector packing. Section 7 presents and analyzes an ad hoc algorithm for the two-dimensional case. Section 8 deals with the results that are related to compact vector summation, and Section 9 gives the discussion.

2 Notation

For a set $X \subseteq [0,1]^d$, we denote by $s(X)$ the sum of all elements of X, and by $D_i(X)$ we denote the i-th component of $s(X)$. Moreover, we define $q(X) = \min_{1\le i\le d} D_i(X)$ to be the smallest component of $s(X)$. Observe that $\textsc{Opt}(X) \le q(X)$ holds.

For $a < b$, we denote by $[a,b]$, (a,b), $(a,b]$, and $[a,b)$ the closed, open and half-open intervals between a and b, respectively. $\mathbb{N}$ is the set of non-negative integers and $\mathbb{R}$ is the set of real numbers. By $\log z$ we denote the base two logarithm of z, and by $\ln z$ the base e logarithm.

3 An Auxiliary Problem: On-line Vector Partitioning

This section deals with an auxiliary problem that is called the *d-dimensional on-line vector partition* problem: A list X of vectors $x^t \in [0,1]^d$, $1 \le t \le n$, has to be partitioned into d parts $P_1, \ldots, P_d$. The vectors $x^t = (x_1^t, \ldots, x_d^t)$ arrive on-line, i.e. when vector x^t arrives it must be assigned to a part P_i before vector x^{t+1} becomes known. The goal is to make the value

$$\min_{1\le i\le d} \; D_i(P_i)/D_i(X) \tag{3}$$

as large as possible. In other words, the i-th part in the partition should get a fair amount of the total size of the i-th coordinate of list X.

For an algorithm A for this partition problem, we denote by $A_1(X), \ldots, A_d(X)$ the partition that A generates for the input list X. To simplify notation, we will sometimes write A_i instead of $A_i(X)$ when the list X is clear from the context. This section is devoted to proving the following theorem.

Theorem 3.1 *For every integer $d \ge 1$ and for every real $0 < \varepsilon < 1$, there exists a positive constant $c_{d,\varepsilon}$ and an on-line algorithm $A = A(d,\varepsilon)$ for the d-dimensional vector partition problem such that*

$$D_i(A_i(X)) \;\ge\; \frac{1-\varepsilon}{d} D_i(X) - c_{d,\varepsilon} \tag{4}$$

holds for all input lists X and for all i, $1 \le i \le d$.

Proof. The proof will be done by induction on the dimension d. For $d = 1$, the claimed algorithm trivially exists (even for $\varepsilon = 0$ and $c_{d,\varepsilon} = 0$). Now assume that the statement of the theorem holds for all dimensions up to $d-1$. Consider some fixed ε, $0 < \varepsilon < 1$.

We introduce a number of technical definitions. Define $\alpha = (d+\varepsilon)/(d+d\varepsilon)$ and note that $\frac{1}{d} < \alpha < 1$. Next, the interval $[0,1]$ is divided into an infinite number of subintervals $\mathcal{I}_j = (\alpha^{j+1}, \alpha^j]$ for $j \in \mathbb{N}$. Moreover, we define $\mathcal{I}_\infty = [0]$. With every vector $c \in (\mathbb{N} \cup \{\infty\})^d$, we associate a subset $\mathcal{C}(c) \subseteq [0,1]^d$ where $x = (x_1, \ldots, x_d) \in \mathcal{C}(c)$ if and only if $x_i \in \mathcal{I}_{c(i)}$ for all $1 \le i \le d$. These sets $\mathcal{C}(c)$ are called *blocks* and constitute an orthogonal partition of the cube $[0,1]^d$ into an infinite number of subsets. Finally, we fix for every vector $c = (c_1, \ldots, c_d) \in (\mathbb{N} \cup \{\infty\})^d$ a *total order* '$\prec$' of its coordinates as follows: For $1 \le i \ne j \le d$, the relation

$i \prec j$ holds if and only if (i) $c_i > c_j$ or (ii) $c_i = c_j$ and $i < j$. Hence, components c_i that are 'small' with respect to $\prec$ correspond to intervals $\mathcal{I}_{c_i}$ that are close to the zero point in $[0,1]$, and components that are 'large' with respect to $\prec$ correspond to intervals that are close to 1.

The algorithm $A = A(d,\varepsilon)$ proceeds as follows. In the background, it simultaneously runs d copies of the algorithm $A(d-1,\varepsilon^2)$ that exists by the inductive assumption. Every background algorithm sees only some fragments of the input list, and it maintains its own private partition of these fragments into $d-1$ parts. Every time a new vector x^t arrives, it is either assigned immediately by the master algorithm $A = A(d,\varepsilon)$, or it is handed over to one of these d background algorithms and then assigned depending on the decision of the background algorithm.

(S0) Let $x^t = (x_1^t, \ldots, x_d^t)$ be the t-th vector that arrives. Determine the vector $c^* \in (\mathbb{N} \cup \{\infty\})^d$ such that x^t belongs to the block $\mathcal{C}(c^*)$. Let k denote the smallest coordinate of c^* with respect to the ordering '$\prec$'.

(S1) If $|\{x^1, \ldots, x^t\} \cap \mathcal{C}(c^*)|$ is divisible by d, then x^t is immediately assigned to the part A_k.

(S2) Otherwise, we remove the k-th component from x^t while keeping the remaining components in the same order. The resulting vector has $d-1$ components and is handed over to the k-th background algorithm who will assign it to, say, its ℓ-th private set in its private partition.

The master algorithm assigns it to the corresponding part while taking care of the removed coordinate: If $\ell < k$ then $A(d,\varepsilon)$ assigns x^t to the part A_ℓ, and otherwise it assigns x^t to the part $A_{\ell+1}$.

Next, we will prove that the algorithm defined above fulfills inequality (4) for $i = 1$. Since the statement of Theorem 3.1 and the algorithm $A(d,\varepsilon)$ are both almost symmetric with respect to all coordinates, this will be sufficient to establish the correctness of Theorem 3.1 (Note that small but harmless asymmetries arise from the definition of the orders '$\prec$').

Define $X(c) = \{x^1, \ldots, x^n\} \cap \mathcal{C}(c)$ and $n(c) = |X(c)|$. For $1 \le i \le d$, let $C^{(i)}$ contain all the vectors $c \in (\mathbb{N} \cup \{\infty\})^d$ for which the i-th coordinate is minimum with respect to '$\prec$'. Let $X^{(i)} = \bigcup_{c \in C^{(i)}} X(c)$.

Let us first deal with blocks $\mathcal{C}(c)$ where $c \in C^{(1)}$. If the first component of c equals j, then by step (S1) above,

$$D_1(A_1 \cap X(c)) \;\ge\; \alpha\frac{1}{d}D_1(X(c)) - \alpha^j \ge \frac{1-\epsilon}{d}D_1(X(c)) - \alpha^j \tag{5}$$

holds, since every d-th vector of $\mathcal{C}(c)$ is assigned to A_1 and since the first component of vectors in $\mathcal{C}(c)$ is at most α^j. It is easy to see that there are only $(j+1)^{d-1}$ vectors $c \in C^{(1)}$ whose first component equals j. By summing up (5) over all vectors in $C^{(1)}$, we derive

$$D_1(A_1 \cap X^{(1)}) \;\ge\; \frac{1-\epsilon}{d}D_1(X^{(1)}) - \sum_{j=0}^{\infty}(j+1)^{d-1}\alpha^j. \tag{6}$$

Elementary calculus shows that the infinite sum in the righthand side of (6) converges for $\alpha < 1$. Hence, it is bounded by some constant depending only on α and d.

Next, we consider blocks $\mathcal{C}(c)$ with $c \in C^{(2)}$. Fix some arbitrary $c \in C^{(2)}$ and assume that its first component equals j. Observe that only $\lfloor n(c)/d \rfloor$ elements of $X(c)$ are assigned to A_2 in step (S1), whereas the remaining $\lceil(1-\frac{1}{d})n(c)\rceil$ elements move on to step (S2). Since $n(c)\alpha^{j+1} \le D_1(X(c)) \le n(c)\alpha^j$, and since the total size of the first components of the vectors packed in step (S1) is at most $\lfloor n(c)/d\rfloor\alpha^j$, the total size $\Delta(c)$ of the first components of the vectors packed in step (S2) fulfills

$$\Delta(c) \;\ge\; n(c)\alpha^{j+1} - \lfloor n(c)/d\rfloor\alpha^j \;\ge\; n(c)\alpha^j\left(\alpha-\frac{1}{d}\right) \;\ge\; \left(\alpha-\frac{1}{d}\right)D_1(X(c)). \tag{7}$$

This yields that the total size of the first components of all vectors in $X^{(2)}$ that move on to step (S2) is at least

$$\sum_{c\in C^{(2)}}\Delta(c) \;\ge\; \left(\alpha-\frac{1}{d}\right)D_1(X^{(2)}). \tag{8}$$

These vectors are packed according to the second background algorithm. Combining the inductive assumption with (8) we derive that

$$\begin{aligned}
D_1(A_1\cap X^{(2)}) \;&\ge\; \frac{1-\varepsilon^2}{d-1}\left(\sum_{c\in C^{(2)}}\Delta(c)\right) - c_{d-1,\varepsilon^2}\\
&\ge\; \frac{1-\varepsilon^2}{d-1}\left(\alpha-\frac{1}{d}\right)D_1(X^{(2)}) - c_{d-1,\varepsilon^2}\\
&=\; \frac{1-\varepsilon}{d}D_1(X^{(2)}) - c_{d-1,\varepsilon^2}.
\end{aligned} \tag{9}$$

For $X^{(3)},\dots,X^{(d)}$, inequalities analogous to (9) can be derived by analogous arguments. Summing up these $d-1$ inequalities for $X^{(2)},\dots,X^{(d)}$, and adding (6) to the result then yields

$$\begin{aligned}
D_1(A_1) \;&=\; D_1(A_1\cap X^{(1)}) + \sum_{i=2}^{d} D_1(A_1\cap X^{(i)})\\
&\ge\; \frac{1-\epsilon}{d}D_1(X^{(1)}) - \sum_{j=0}^{\infty}(j+1)^{d-1}\alpha^j + \frac{1-\varepsilon}{d}\sum_{i=2}^{d} D_1(X^{(i)}) - (d-1)c_{d-1,\varepsilon^2}\\
&\ge\; \frac{1-\varepsilon}{d}\sum_{i=1}^{d} D_1(X^{(i)}) - c_{d,\varepsilon}\\
&=\; \frac{1-\varepsilon}{d}D_1(X) - c_{d,\varepsilon}
\end{aligned}$$

where $c_{d,\varepsilon} \ge (d-1)c_{d-1,\varepsilon^2} + \sum_{j=0}^{\infty}(j+1)^{d-1}\alpha^j$ is a constant. This proves inequality (4) for $i=1$ and also completes the proof of Theorem 3.1. ∎

What about the time complexity of algorithm $A(d,\varepsilon)$? It is straightforward to get a running time of $O(n)$, where the constant hidden in the O-notation depends on d and ε.

4 An On-line Algorithm for Vector Covering

In this section, we describe on-line approximation algorithms for vector covering whose worst case ratio comes arbitrarily close to $1/(2d)$. Let us first recapitulate the corresponding result for $d=1$.

Observation 4.1 *(Assmann, Johnson, Kleitman, Leung [3])*
There exists an on-line approximation algorithm G for one-dimensional vector covering that fulfills

$$G(X) \ \geq \ D_1(X)/2 - 1 \ \geq \ \mathrm{OPT}(X)/2 - 1 \tag{10}$$

for all input lists X. Hence, $R_G = \frac{1}{2}$.

Proof. The *greedy algorithm* G (that covers bins one by one) puts into every bin a set of items with overall size at most 2. ■

Theorem 4.2 *For every integer $d \geq 2$ and for every real $0 < \varepsilon < 1$, there exists an on-line algorithm $B = B(d, \varepsilon)$ for the d-dimensional vector covering problem with asymptotic worst case ratio $R_B = (1 - \varepsilon)/(2d)$.*

Proof. For every input list X, algorithm $B(d, \varepsilon)$ simulates the algorithm $A(d, \varepsilon)$ for on-line vector partitioning from Theorem 3.1 and uses the resulting partition $A_1, \ldots, A_d$. If $A(d, \varepsilon)$ puts item x^t into the i-th part A_i, algorithm $B(d, \varepsilon)$ forgets about all coordinates of x^t with the exception of the i-th coordinate, and thus prunes x^t down to a one-dimensional item.

By applying the greedy algorithm from Observation 4.1, item x^t then is used for covering the i-th coordinate of the bins. Therefore, for every coordinate $1 \leq i \leq d$, algorithm B will cover the i-th component of at least $(1 - \varepsilon)D_i(X)/(2d) - O(1)$ unit bins. This implies that

$$B(X) \ \geq \ \min_{1 \leq i \leq d} \frac{1 - \varepsilon}{2d} D_i(X) - O(1) \ \geq \ \frac{1 - \varepsilon}{2d} \mathrm{OPT}(X) - O(1), \tag{11}$$

since $\mathrm{OPT}(X) \leq \min_{1 \leq i \leq d} D_i(X)$ holds. This completes the proof. ■

5 A Lower Bound for On-line Vector Covering Algorithms

In this section, we show that for $d \geq 2$ there does not exist an on-line algorithm with worst case ratio better than $2/(2d + 1)$. Note that this lower bound is only a factor of 2 away from the upper bound $1/(2d)$ that has been derived in the preceding section.

Let us suppose that there exists a dimension $d \geq 2$ and that there exists an on-line algorithm A for d-dimensional vector covering that fulfills

$$R_A \ \geq \ \frac{2}{2d + 1} + \tau \qquad \text{for some } \tau > 0. \tag{12}$$

We are going to derive a contradiction from this. Inequality (12) implies that there exists a threshold $k^{(\tau)}$ such that

$$\frac{2}{2d + 1} + \frac{\tau}{2} \ \leq \ A(X)/\mathrm{OPT}(X), \qquad \text{whenever} \quad \mathrm{OPT}(X) \geq k^{(\tau)} \tag{13}$$

holds, where X is any finite list over $[0, 1]^2$. Consider a sufficiently large value k that fulfills $k \geq k^{(\tau)}$ (the precise value of k will be determined later), and define a small positive real number $\varepsilon = 1/(4k^d)$. Define $d + 1$ homogeneous lists X_1, X_2, X_3 and Y_i, $1 \leq i \leq d - 2$, as follows.

- List X_1 contains $2k^d$ times the vector $(1 - \varepsilon, \frac{1}{k}, \frac{1}{k^2}, \frac{1}{k^3}, \ldots, \frac{1}{k^{d-1}})$.

- List X_2 contains $2k^d$ times the vector $(\varepsilon, 1, 1, 1, \ldots, 1)$.
- List X_3 contains k^d times the vector $(0, 1, 1, 1, \ldots, 1)$.
- For $1 \leq i \leq d-2$, list Y_i contains $2k^{i+1}$ times the vector $(\underbrace{0, \ldots, 0}_{(d-i)-times}, \underbrace{1, \ldots, 1}_{i-times})$.

Next, we feed list X_1 to algorithm A and we investigate the resulting partitioning of X_1 into bins. Let z_1 denote the number of bins that contain exactly 1 item, and let z_2 denote the number of bins that contain at least 2 but less than k items. For $1 \leq j \leq d-2$, let m_j denote the number of bins that contain at least k^j but less than k^{j+1} items. Finally, m_{d-1} denotes the number of bins that contain at least k^{d-1} items. Clearly,

$$z_1 + 2z_2 + \sum_{j=1}^{d-1} k^j m_j \ \leq \ 2k^d \tag{14}$$

holds, and this implies

$$\sum_{j=1}^{d-1} m_j \ \leq \ 2k^{d-1}. \tag{15}$$

Next, we observe that $A(X_1) = m_{d-1} \leq 2k$ and $\text{OPT}(X_1) = 2k$ holds. By (13),

$$\frac{2}{2d+1} + \frac{\tau}{2} \ \leq \ \frac{m_{d-1}}{2k}. \tag{16}$$

Moreover, $A(X_1X_2) \leq z_1 + z_2 + \sum_{j=1}^{d-1} m_j$, since A cannot fill any bin with items from X_2 if the bin does not contain at least one item from X_1. Together with $\text{OPT}(X_1X_2) = 2k^d$ and the inequality in (13), this leads to

$$\frac{2}{2d+1} + \frac{\tau}{2} \ \leq \ \frac{1}{2k^d}(z_1 + z_2 + \sum_{j=1}^{d-1} m_j). \tag{17}$$

Next, observe that $A(X_1X_3) \leq z_2 + \sum_{j=1}^{d-1} m_j$ since the first coordinates of the items in X_3 are too small to fill a bin that does not already contain at least two items from X_1. Since $\text{OPT}(X_1X_3) = k^d$, we get from (13) that

$$\frac{2}{2d+1} + \frac{\tau}{2} \ \leq \ \frac{1}{k^d}(z_2 + \sum_{j=1}^{d-1} m_j). \tag{18}$$

Next, observe that for $1 \leq i \leq d-2$, $\text{OPT}(X_1Y_i) = 2k^{i+1}$, whereas $A(X_1Y_i) \leq \sum_{j=d-i-1}^{d-1} m_j$. Hence for $1 \leq i \leq d-2$, inequality (13) yields

$$\frac{2}{2d+1} + \frac{\tau}{2} \ \leq \ \frac{1}{2k^{i+1}}(\sum_{j=d-i-1}^{d-1} m_j). \tag{19}$$

Now we multiply (18) by $\frac{1}{2}$, and add the inequalities (16) and (17) and the $d-2$ inequalities (19) for $1 \le i \le d-2$ to it. This yields

$$1+\frac{\tau}{4}(2d+1) \le \frac{1}{2k^d}\left(z_1+2z_2+2\sum_{j=1}^{d-1} m_j+\sum_{i=0}^{d-2} k^{d-i-1}\sum_{j=d-i-1}^{d-1} m_j\right) \tag{20}$$

$$= \frac{1}{2k^d}\left(z_1+2z_2+2\sum_{j=1}^{d-1} m_j+\frac{k}{k-1}\sum_{j=1}^{d-1} k^j m_j\right) \tag{21}$$

$$\le \frac{1}{2k^d}\left(2k^{d-1}+\frac{k}{k-1}2k^d\right) = \frac{1}{k}+\frac{k}{k-1}. \tag{22}$$

where we applied (14) and (15) in order to derive the last inequality. As k tends to infinity, the righthand side $1/k+k/(k-1)$ in (22) tends to 1, whereas the lefthand side is bounded strictly away from 1. This is a contradiction, and hence an on-line algorithm A that fulfills (12) cannot exist.

Theorem 5.1 *For any on-line algorithm A for d-dimensional vector covering with $d \ge 2$, the inequality $R_A \le 2/(2d+1)$ holds.* ■

A simple extension of the above argument combined with Yao's theorem [15] yields an analogous result for *randomized* on-line algorithms. We omit the straightforward details.

Theorem 5.2 *For any* randomized *on-line algorithm A for d-dimensional vector covering with $d \ge 2$, the inequality $R_A \le 2/(2d+1)$ holds.* ■

6 Off-line Results for Vector Covering

In this section, we first provide a *combinatorial* lower bound for the size of the optimum solution of d-dimensional vector covering problems. This lower bound is based on a probabilistic argument and then yields (by applying derandomization) an off-line approximation algorithm with worst case guarantee $\Omega(1/\log d)$.

Theorem 6.1 *For $d \ge 2$, let $\mu_d = 2\ln d + 2\ln\ln d + 2$. Then for every set $X \subseteq [0,1]^d$ with $q(X) \ge \mu_d$,*

$$\textsc{Opt}(X) \ge \left\lfloor\frac{q(X)}{\mu_d}\right\rfloor\left(1-\frac{1}{\ln d}\right) = \left\lfloor\frac{q(X)}{2\ln d}\right\rfloor(1-o(1)). \tag{23}$$

Proof. Color the vectors in X with $\lfloor q(X)/\mu_d\rfloor \ge 1$ colors, where each vector is colored randomly and independently according to a uniform distribution on the colors. Fix a color c^* and consider the sum Ψ of all vectors with this color c^* at a fixed coordinate. The expected value μ of this sum is at least μ_d.

By a standard estimate (cf., e.g., the remark after Theorem 2 in [11]) the following result holds for a weighted sum Ψ of independent Bernoulli trials where all weights are real numbers in $[0,1]$: If μ is the expected value of Ψ and if $0 < \gamma \le 1$, then

$$\Pr[\Psi-\mu < -\gamma\mu] < \exp(-\gamma^2\mu/2). \tag{24}$$

By setting $\gamma = 1 - 1/\mu$ in our case, we derive from this that the probability that the sum of vectors at the fixed coordinate is smaller than one is

$$\begin{aligned}\Pr[\Psi < 1] &= \Pr[\Psi - \mu < -\gamma\mu] \;<\; \exp(-\gamma^2\mu/2) \\ &< \exp(1-\mu/2) \;<\; \exp(-\ln d - \ln\ln d) \;=\; \frac{1}{d\ln d}.\end{aligned}$$

Therefore, the probability that the sum of vectors with color c^* will be less than 1 in some coordinate does not exceed $d \cdot 1/(d\ln d) = 1/\ln d$. It follows that the expected number of colors whose elements form a unit cover is at least $(1 - \frac{1}{\ln d})\lfloor q(X)/\mu_d \rfloor$. ■

We remark that up to a constant factor, the lower bound in (23) is best possible. This can be deduced from one of the results in [1]. It is proved there (see Section 3) that for every d there is a collection $\mathcal{F}$ of d subsets of cardinality d each of a set Z of size $\lfloor d\ln d \rfloor$ so that no set of at most $\ln^2 d - 10\ln d$ members of Z intersects all subsets in the collection. Associate each element z of Z with a $(0,1)$-vector $v_z = (v_z(F) : F \in \mathcal{F})$ of length d whose coordinates are indexed by the subsets in $\mathcal{F}$, where $v_z(F) = 1$ if $z \in F$ and $v_z(F) = 0$ otherwise. Let X be the set of all these vectors. Every coordinate of the sum of the members of X is precisely d, that is $q(X) = d$. Also, by the properties of the above collection, every unit cover must contain at least $\ln^2 d - 10\ln d$ vectors. Therefore

$$\textsc{Opt}(X) \;\leq\; \frac{\lfloor d\ln d \rfloor}{\ln^2 d - 10\ln d} = (1+o(1))\frac{d}{\ln d} = (1+o(1))\frac{q(X)}{\ln d}, \tag{25}$$

proving the remark. Note also that by duplicating each vector as many times as needed, similar examples X exist in which the value of $\textsc{Opt}(X)$ is arbitrarily large.

Moreover, by applying Raghavan's method of conditional probabilities with pessimistic estimators [11], we can convert the probabilistic existence proof of Theorem 6.1 into a deterministic polynomial time algorithm. Since $\textsc{Opt}(X) \leq q(X)$, this yields the following theorem.

Theorem 6.2 *For every $d \geq 2$, there exists a deterministic polynomial time off-line approximation algorithm for d-dimensional vector covering with asymptotic worst case ratio $(1+o(1))/(2\ln d)$.* ■

7 A Simple Off-line Algorithm for Dimension Two

In this section, we present a fast and simple off-line approximation algorithms for $d = 2$. It will produce an approximative solution that covers at least $q(X)/2 \geq \textsc{Opt}(X)/2$ bins.

The algorithm assumes that the input list X fulfills $q(X) = D_1(X) = D_2(X)$ (otherwise, decrease some of the coordinates to make these values equal). Moreover, the algorithm sometimes make use of a *garbage bin*: The items in X will be classified into four classes. In case that one of these classes contains only a single item, this item is thrown into the garbage bin and disregarded from further arguments (in the end, the contents of the garbage bin is merged with one part of the constructed partition of X). Since there are just four classes, the garbage bin will contain at most four items and it will not effect the asymptotic worst case ratio of our algorithm.

Next, let us classify the items: Items with two large components in the interval $(\frac{1}{2}, 1]$ are called of type $(+,+)$. Items where only the first component is large are called of type $(+,-)$, items where only the second component is large are called of type $(-,+)$, and items with both components in the interval $[0, \frac{1}{2}]$ are called of type $(-,-)$.

The algorithm B_2 goes through the following four steps (0)—(3).

> (Step 0). As long as there exist two items x and y of type $(-,-)$, replace them by a new item $x+y$. If there is a single item of type $(-,-)$, it is thrown into the garbage bin.

From now on, all items have at least one coordinate that is strictly greater than $\frac{1}{2}$.

> (Step 1). While there are at least two items of type $(+,-)$ and at least two items of type $(-,+)$, repeat the following step: Take x of type $(+,-)$ and y of type $(-,+)$. If $x+y \leq (1,1)$, replace them by a new item $x+y$ of type $(+,+)$. If $x+y \geq (1,1)$, pack them together and produce a unit cover. Otherwise assume w.l.o.g. that the first coordinate of $x+y$ is in $[1, \frac{3}{2})$ and the other coordinate is in $(\frac{1}{2}, 1)$. Add an arbitrary item of type $(-,+)$ and close the corresponding (covered!) bin.

Without loss of generality, we assume that we eventually run out of $(+,-)$-items (a single surviving item of this type is thrown into the garbage bin) and that there only remain items of type $(-,+)$ and $(+,+)$. In case we eventually run out of $(-,+)$-items, we apply a symmetric version of (Step 3) below.

> (Step 2). While there are at least two items of type $(+,+)$, the sum of these two items has both coordinates in the interval $(1,2]$. We pack such a pair together to cover a unit bin. If in the end a single item of type $(+,+)$ remains, it is thrown into the garbage bin.
>
> (Step 3). Finally, pack the remaining items of type $(-,+)$ with the greedy algorithm according to the first coordinate.

Lemma 7.1 *The asymptotic worst case ratio of the off-line approximation algorithm B_2 defined via the above three steps equals $\frac{1}{2}$. The algorithm can be implemented to run in $O(n)$ time.*

Proof. All bin covers that are produced during (Step 1) and (Step 2) have *both* components in $[1,2]$. By Observation 4.1, the first component of all bin covers produced in (Step 3) is at most 2. Hence, the number of produced bin covers is at least $q(X)/2 - 5/4$, where the additive constant accounts for the items in the garbage bin.

This yields that the worst case ratio is at least $\frac{1}{2}$. That this bound is tight follows by considering the list X^* containing n-times the item $(1-\varepsilon, \frac{1}{2}+\varepsilon)$ and n-times the item $(3\varepsilon, \frac{1}{2}+\varepsilon)$, where $\varepsilon < 1/(3n)$ is some small positive real. Here $\text{OPT}(X^*) = n$ and $B_2(X^*) \leq \frac{n}{2}+1$. Reaching the claimed time complexity is straightforward. ■

8 An Off-line Algorithm with Worst Case Ratio 1/d

In this section, we present an off-line approximation algorithm that exploits a method from the area of compact vector summation (see Sevastianov [13, 14], and also Beck and Fiala [4].) The approximation algorithm has a worst case ratio of $1/d$ and is essentially based on the following proposition.

Proposition 8.1 *(Sevastianov [13])*
Let $W = \{w_1, \dots, w_n\} \subseteq \mathbb{R}^d$, *let* $\delta_i' \in [0,1]$, $1 \le i \le n$ *and let* $w^* = \sum_{i=1}^n \delta_i' w_i$. *Then one can find in* $O(nd^2)$ *time a set of reals* $\delta_i \in [0,1]$ *such that*

- $\sum_{i=1}^n \delta_i w_i \;=\; w^*$, *and*
- $|\{i \mid 0 < \delta_i < 1\}| \;\le\; d.$ ■

In other words, the linear dependence $w^* = \sum_{i=1}^n \delta_i w_i$ can be easily transformed into another linear dependence where almost all of the coefficients are 0 or 1.

Let $\vec{1}$ denote the vector that has all of its d components equal to 1. For a vector $x \in \mathbb{R}^d$ and an integer i, $1 \le i \le d$, we will denote by $x(i)$ the i-th component of vector x.

Lemma 8.2 *For an integer* $d \ge 2$ *and for a family* $\{x_1, \dots, x_n\} \subset [0,1]^d$, *of vectors, denote* $z \doteq s(X)/q(X) \ge \vec{1}$. *Then we can find in* $O(nd^2)$ *time a subset* $X' \subseteq X$ *such that*

$$\vec{1} \;\le\; z \;\le\; s(X') \;\le\; dz. \tag{26}$$

Proof. Since for coefficients $\delta_j' = 1/q(X)$, $1 \le j \le n$, the equation $z = \sum_{j=1}^n \delta_j' x_j$ holds, we may apply Proposition 8.1 to transform the numbers $\{\delta_j'\}$ into reals $\delta_j \in [0,1]$ so that

$$\sum_{j=1}^n \delta_j x_j = z, \tag{27}$$

$$|\{j \mid \delta_j \in (0,1)\}| \le d. \tag{28}$$

We denote $N' = \{j \mid \delta_j \in (0,1)\}$ and $N'' = \{j \mid \delta_j = 1\}$, and we define $y = \sum_{j \in N'' \cup N'} x_j$ and $u = \sum_{j \in N''} x_j$. Then we have by these definitions that

$$y \ge \sum_{j \in N'' \cup N'} \delta_j x_j = z \ge \sum_{j \in N''} x_j = u. \tag{29}$$

If $y \le z + (d-1)\vec{1}$, then $y \le dz$, and we are done since the x_j with indices in $N'' \cup N'$ form the desired solution set. Hence, from now on we assume without loss of generality that

$$y(1) > z(1) + d - 1. \tag{30}$$

Since $u(1) \le z(1)$ and $y(1) = u(1) + \sum_{j \in N'} x_j(1) > z(1) + d - 1$ holds, we obtain

$$u(1) + x_j(1) \;>\; z(1) \qquad \text{for all } j \in N'. \tag{31}$$

If $\sum_{j \in N'} \delta_j \ge 1$ then

$$y(1) - z(1) \;=\; \sum_{j \in N'} (1-\delta_j) x_j(1) \;\le\; \sum_{j \in N'} (1-\delta_j) \;=\; |N'| - \sum_{j \in N'} \delta_j \;\le\; d-1, \tag{32}$$

which contradicts (30). Hence,

$$\sum_{j \in N'} \delta_j \;<\; 1. \tag{33}$$

Next, let us prove that

$$\forall \nu \; \exists j_\nu \in N' : \quad u(\nu) + x_{j_\nu}(\nu) \geq z(\nu). \tag{34}$$

Suppose, that this is not the case and that for some coordinate ν, $1 \leq \nu \leq d$, we have $u(\nu) + x_j(\nu) < z(\nu)$ for all $j \in N'$. But in this case

$$z(\nu) \; = \; u(\nu) + \sum_{j \in N'} \delta_j x_j(\nu) \; < \; u(\nu) + \sum_{j \in N'} \delta_j \left(z(\nu) - u(\nu) \right) \; < \; z(\nu), \tag{35}$$

where the last inequality follows from (33). This contradiction proves (34).

Finally, let us define $\tilde{N}$ to be the set of indices $\{j_2, \ldots, j_d\}$ that exist by (34). Clearly, $|\tilde{N}| \leq d - 1$. It follows from (31) and (34) that

$$u(\nu) + \sum_{j \in \tilde{N}} x_j(\nu) \; \geq \; z(\nu) \qquad \text{for all } \nu = 1, \ldots, d. \tag{36}$$

Thus, for $y' \doteq \sum_{j \in N'' \cup \tilde{N}} x_j$, we have $y' \geq z$. Moreover,

$$y'(\nu) - z(\nu) \; \leq \; y'(\nu) - u(\nu) \; = \; \sum_{j \in \tilde{N}} x_j(\nu) \; \leq \; d - 1 \tag{37}$$

holds for all $\nu = 1, \ldots, d$, which implies $z \leq y' \leq z + (d-1)\vec{1} \leq dz$. Hence, $X' = \{x_i | i \in N'' \cup \tilde{N}\}$ yields the required set. ■

Theorem 8.3 *For every $d \geq 2$, there exists a deterministic polynomial time off-line approximation algorithm for d-dimensional vector covering with asymptotic worst case ratio $1/d$. The algorithm can be implemented to run in $O(d^2 n^2)$ time.*

Proof. Let $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ be an input list for d-dimensional vector covering. We apply the algorithm described in Lemma 8.2 repeatedly to list X, and remove in every step the corresponding set X' from X. Clearly, every step produces a unit cover, and there are at least $\lfloor q(X)/d \rfloor$ steps. The claimed time complexity follows from Proposition 8.1. ■

9 Discussion

In this paper, we derived the first non-trivial approximation algorithms for on-line and off-line vector covering. There remain many open questions.

(Q1) Determine the exact approximability threshold for the on-line version! We feel that our upper bound $1/(2d)$ should be closer to the true approximability threshold than our lower bound $2/(2d+1)$.

(Q2) Find lower bounds for the off-line version! We do not know of any non-approximability results for polynomial time approximation algorithms for the vector covering problem. Can our $\Omega(1/\log d)$ approximability result be beaten asymptotically? Does the problem allow an approximation algorithm whose worst case ratio is a constant (that does not depend on the dimension d)? Does it allow a polynomial time approximation scheme?

(Q3) Is it possible to improve on the ancient $\frac{3}{4}$ approximation algorithm of Assmann et al [3] for the one-dimensional bin covering problem? This seems to be very difficult. Can one at least disprove the existence of a polynomial time approximation scheme?

References

[1] N. Alon, Transversal numbers of uniform hypergraphs, *Graphs and Combinatorics* **6**, 1990, 1–4.

[2] S.F. Assmann, "Problems in Discrete Applied Mathematics", Doctoral Dissertation, Mathematics Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1983.

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

[4] J. Beck and T. Fiala, Integer-making Theorems, *Disc. Appl. Math.* **3**, 1981, 1–8.

[5] J. Csirik and J.B.G. Frenk, A dual version of bin packing, *Algorithms Review* **1**, 1990, 87–95.

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

[7] J. Csirik and V. Totik, On-line algorithms for a dual version of bin packing, *Discr. Appl. Math.* **21**, 1988, 163–167.

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

[9] T. Gaizer, An algorithm for the 2D dual bin packing problem, unpublished manuscript, University of Szeged, Hungary, 1989.

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

[11] P. Raghavan, Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs, *Journal of Computer and System Sciences* **37**, 1988, 130–143.

[12] M.B. Richey, Improved bounds for harmonic-based bin packing algorithms, *Discr. Appl. Math.* **34**, 1991, 203–227.

[13] S.V. Sevastianov, Geometry in the theory of scheduling, *Trudy Instituta Matematiki Sibirskogo Otdelenia Akademii Nauk SSSR* **10**, 1988, 226–261. (in Russian).

[14] S.V. Sevastianov, On some geometric methods in scheduling theory: a survey, *Discr. Appl. Math.* **55**, 1994, 59–82.

[15] A.C.C. Yao, Probabilistic Computations: Towards a unified measure of complexity, in *Proceedings of the 18th ACM Symposium on Theory of Computing*, 1977, 222-227.