Available online at www.sciencedirect.com
SCIENCE DIRECT®
Operations Research Letters 31 (2003) 35–41
Operations Research Letters
www.elsevier.com/locate/dsw

An approximation algorithm with absolute worst-case performance ratio 2 for two-dimensional vector packing

Hans Kellerer[a,*], Vladimir Kotov[b]

[a] *Institut für Statistik und Operations Research, Universität Graz, Universitätsstraße 15, A-8010 Graz, Austria*
[b] *Faculty of Applied Mathematics and Computer Science, Byelorussian State University, Minsk 220050, Byelorussia*

Received 24 April 2002; received in revised form 26 May 2002; accepted 30 May 2002

Abstract

The two-dimensional vector packing problem is the generalization of the classical one-dimensional bin packing problem to two dimensions. While an asymptotic polynomial time approximation scheme has been designed for one-dimensional bin packing, the existence of an asymptotic polynomial time approximation scheme for two dimensions would imply $P=NP$. The existence of an approximation algorithm for the two-dimensional vector packing problem with an asymptotic performance guarantee 2 was an open problem so far. In this paper we present an $O(n\log n)$ time algorithm for two-dimensional vector packing with absolute performance guarantee 2.

Keywords: Bin packing; Vector packing; Worst-case analysis

1. Introduction

In the classical one-dimensional *bin packing problem* we are given a set of items with weights in $[0,1]$ and an infinite number of unit-capacity bins. The goal is to pack the items into a minimum number of bins such that the total weight of the items in each bin does not exceed the bin-capacity one. The *d-dimensional vector packing problem* is the generalization of one-dimensional bin packing to d dimensions. Instead of being a single number, in the d-dimensional vector packing problem each item is a d-dimensional vector with coordinates in $[0,1]$. The goal is to pack all items into a minimum number of bins provided with d corresponding capacities all equal to one. Not only packing problems can be modeled by vector packing problems but also scheduling problems with resource constraints (see [5]).

It is well-known that bin packing and vector packing (as a generalization) are *NP*-hard in the strong sense. No polynomial time approximation algorithm with a worst-case performance ratio better than $\frac{3}{2}$ can exist for bin packing, unless $P=NP$. Fernandez de la Vega and Lueker [4] gave an *asymptotic polynomial time approximation scheme* (APTAS) for bin packing. They extended their method to an asymptotic $(d+\varepsilon)$-approximation algorithm for d-dimensional vector packing where ε can be chosen arbitrarily close to zero. Recently, Chekuri and Khanna [3] improved this result for arbitrary $\varepsilon>0$ proposing a polynomial-time algorithm with worst-case performance ratio $1+d/k+H_{k-1}$ where $k=\lceil 1/\varepsilon\rceil$ and $H_k := 1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{k}$ denotes the kth harmonic number. Their approach is based on

* Corresponding author.
E-mail addresses: hans.kellerer@uni-graz.at (H. Kellerer), kotov@fpm.bsu.unibel.by (V. Kotov).

0167-6377/03/$ - see front matter
PII: S0167-6377(02)00173-6

a linear programming relaxation of the problem. The existence of an *APTAS*, even for the case $d = 2$, was ruled out by Woeginger [8] and (independently) by Chekuri and Khanna [3] (on the assumption $P \neq NP$).

In this paper we will address the *two-dimensional vector packing problem* (2-DVPP). Items $b = (b^x, b^y)$ are two-dimensional vectors with x-coordinate and y-coordinate. They have to be packed in a minimum number of bins with two capacities each equal to one.

Vercruyssen and Muller [7] gave an application of two-dimensional vector packing in the steel industry, an application in the logistics of cargo airplanes is mentioned in [1]. Spieksma [6] introduced a branch-and-bound algorithm for the 2-DVPP.

The existence of an approximation algorithm for the 2-DVPP with an asymptotic performance guarantee 2 was an open problem so far. We will present an approximation algorithm with $O(n \log n)$ running time for the 2-DVPP with an *absolute* performance guarantee 2. Note that the linear programming approach by Chandra and Chekuri yields only an approximation algorithm with asymptotic worst-case performance ratio three for the 2-DVPP. Note also, that the $(2 + \varepsilon)$-approximation algorithm by Fernandez de la Vega and Lueker for the 2-DVPP runs in time $O(n) + D_\varepsilon$ where D_ε can exceed

$$\begin{pmatrix} \lceil (1 + \frac{2}{\varepsilon})^2 \rceil \\ \lfloor 1 + \frac{2}{\varepsilon} \rfloor \end{pmatrix}.$$

At first sight it seems easy to construct a heuristic for the 2-DVPP with asymptotic worst-case performance ratio 2. For evaluating algorithms with respect to their worst-case performance, methods for calculating good lower bounds of the number of bins in the optimal solution are necessary. Obvious lower bounds for the 2-DVPP are the maximum number of items with x-coordinate greater than $\frac{1}{2}$ (or y-coordinate greater than $\frac{1}{2}$, respectively) and the total sum of the x-coordinates of the items (or the total sum of the y-coordinates of the items, respectively).

By using only these standard lower bounds it is not possible to prove that a heuristic has asymptotic worst-case performance ratio 2. This can be illustrated by the following example. Let LB denote a lower bound on the number of bins which are necessary to pack all items of a 2-DVPP instance and C^* the number of bins in an optimal solution. We are given n items $(1 - \varepsilon, 0)$ and $3n$ items $(2\varepsilon, 1/3 + \varepsilon)$ with ε an arbitrary small number greater than zero. From the lower bounds above we get $LB > n$ but it can be easily seen that the optimal solution needs $\frac{5}{2}n$ bins to pack all items.

Caprara and Toth [2] introduced a compatibility graph associated with a 2-DVPP instance where two items are called compatible if they can be packed into the same bin, and incompatible otherwise. Of course, the maximum cardinality of an independent set in this graph is a lower bound on the optimal solution value. Caprara and Toth prove that in this case there are instances with $LB/C^* \geqslant 5/2$. They investigated also the linear programming relaxation of the integer linear programming formulation of the 2-DVPP. Also this approach is not very promising since they could show that for this lower bound there are instances with $LB/C^* \geqslant 3$. Moreover, combinations of all these bounds did not improve the results.

Our elementary algorithm tries to avoid the problems mentioned above. It enables us to recalculate non-trivial lower bounds for the optimal solution during the execution of the algorithm when necessary.

2. Algorithm and analysis

In the beginning of this section we will introduce some further notations which are used in the rest of the paper. For a set of items S the total sum of the x-coordinates (y-coordinates) of the items in S is called the *x-weight of S* (*y-weight of S*) and is denoted by $W^x(S)$ ($W^y(S)$). The weight of a set of bins corresponds to the total weight of the items contained in that bins.

Items $b = (b^x, b^y)$ with $b^x > 1/2$ and $b^y > 1/2$ are collected in the set *BigXY* and called *BigXY*-items. (If a property holds for both the x-coordinate and the y-coordinate, we will use the index x, y instead of x or y. So, *BigXY*-items can also be described by the property $b^{x,y} > \frac{1}{2}$ instead of $b^x > \frac{1}{2}$ and $b^y > \frac{1}{2}$.) Items $b = (b^x, b^y)$ with $b^x > \frac{1}{2}$ and $b^y \leqslant \frac{1}{2}$ are collected in the set *BigX* and called *BigX*-items. Analogously, items $b=(b^x, b^y)$ with $b^x \leqslant \frac{1}{2}$ and $b^y > \frac{1}{2}$ are collected in the set *BigY* and called *BigY*-items. Items with at least one coordinate greater than $\frac{1}{2}$, are called *big* items. All the other items are called *small* items. Especially, $Small(x \geqslant y)$ contains the small items with

the x-coordinate not smaller than the y-coordinate, i.e. for each $s \in Small(x \geqslant y)$ holds $s^x \geqslant s^y$. Analogously, $Small(y > x)$ consists of all small items s with $s^y > s^x$.

Set $n := |BigX|$ and $m := |BigY|$. W.l.o.g. assume that $n \geqslant m$. The $BigX$-items are denoted by $v_1, \ldots, v_n$ and are sorted in a list in non-increasing order of the x-coordinates, i.e. $v_1^x \geqslant v_2^x \geqslant \cdots \geqslant v_n^x$. The $BigY$-items are denoted by $w_1, \ldots, w_m$ and are sorted in a list in non-increasing order of the y-coordinates, i.e. $w_1^y \geqslant w_2^y \geqslant \cdots \geqslant w_m^y$.

Let b be a big item. Then, Add_b is the set of all small items s which fit in a bin together with item b, i.e. $s^{x,y} + b^{x,y} \leqslant 1$ for all $s \in Add\_b$. If b is a $BigX$-item, assume the items in Add_b to be sorted in a list in non-increasing order of the ratio of the y-coordinate to the x-coordinate, i.e. $s_1^y/s_1^x \geqslant s_2^y/s_2^x$ and so on. Otherwise, if b is a $BigY$-item, assume the items in Add_b to be sorted in a list in non-increasing order of the ratio of the x-coordinate to the y-coordinate. $Add\_b(x \geqslant y)$ is the set of all items $s \in Add\_b$ with $s^x \geqslant s^y$. Analogously, $Add\_b(y > x)$ is the set of all items $s \in Add\_b$ with $s^y > s^x$. The set Not_Add_b contains all small items s which cannot be put into a bin with big item b, i.e. $s^x + b^x > 1$ or $s^y + b^y > 1$ holds. $Not\_Add\_b(x \geqslant y)$ consists of all items $s \in Not\_Add\_b$ with $s^x \geqslant s^y$. Analogously, $Not\_Add\_b(y > x)$ consists of all items $s \in Not\_Add\_b$ with $s^y > s^x$.

For the description of the algorithm we will collect bins in so-called *bunches*. If we have a bunch consisting of i bins $B_1, \ldots, B_i$, we speak of a $BU(i)$ *bunch*. We call a $BU(i)$ bunch BU *x-complete* if $W^x(BU) > i/2$, otherwise *x-incomplete*. A $BU(i)$ bunch BU is called *y-complete* if $W^y(BU) > i/2$, otherwise *y-incomplete*. If BU is both x-complete and y-complete, it is called *complete*.

Now we turn to a detailed description of our algorithm. We first explain procedure $P(\cdot)$ which is called several times during our main algorithm. Here the $\cdot$ stands for a $BigX$-item v or for a $BigY$-item w.

Procedure P(v). Given a $BU(2)$ bunch consisting of a $BigX$-item v in bin B_1 and a second bin B_2 (containing possibly a $BigY$-item), procedure $P(v)$ puts each item of $NotAdd\_v(y > x)$ which fits, into bin B_2. Then, the items of the list of Add_v are assigned to B_2 (in non-increasing order of the ratio of the y-coordinate to the x-coordinate). If there is an item which does not fit into B_2, it is put into B_1 and the filling of the bunch is stopped. If all items of Add_v can be assigned, the bunch is called *y-semicomplete*.

Procedure P(w). For a $BU(2)$ bunch consisting of a bin B_1 (containing possibly a $BigX$-item) and a $BigY$-item w in a second bin B_2, procedure $P(w)$ puts each item of $NotAdd\_w(x \geqslant y)$ which fits, into bin B_1. Then, the items of the list of Add_w are assigned to B_1 (in non-increasing order of the ratio of the x-coordinate to the y-coordinate). If there is an item which does not fit into B_1, it is put into B_2 and the filling of the bunch is stopped. If all items of Add_w can be assigned, the bunch is called *x-semicomplete*.

The description of the main algorithm is split into several parts. In Step 1 all $BigXY$-items are put into separate bins. Then the actual first items from the list of sorted items of $BigX$ and $BigY$, respectively, form bunches in Step 2. In Step 2.1, a $BU(1)$ bunch is created if the $BigX$-item and the $BigY$-item fit together in a bin and a $BU(2)$ bunch, otherwise. Obviously, the $BU(1)$ bunches are complete. If a $BU(2)$ bunch is complete, no other items are added (Step 2.2). If a $BU(2)$ bunch is not complete, we add small items by applying Procedure $P(\cdot)$ in Step 2.3. After running out of the $BigY$-items, bunches are formed with the remaining $BigX$-items. This is described in Step 3. Formally, the first three steps of the algorithm read as follows:

1. Put all items of $BigXY$ into $BU(1)$ bunches.
2. Consider the two lists $(v_1, v_2, \ldots, v_m)$ and $(w_1, w_2, \ldots, w_m)$ of sorted items of $BigX$ and $BigY$, respectively. Remove the (actual) first items from each list, $v \in BigX$ and $w \in BigY$.
   - 2.1. If $v^{x,y} + w^{x,y} \leqslant 1$, put them into a $BU(1)$ bunch.
   - 2.2. If $v^{x,y} + w^{x,y} > 1$, put v into a bin B_1 and w into a bin B_2 and form a $BU(2)$ bunch.
   - 2.3. Put v into bin B_1 and w into bin B_2 of a $BU(2)$ bunch. If $v^x + w^x > 1$ and $v^y + w^y \leqslant 1$, call procedure $P(v)$. Otherwise if $v^x + w^x \leqslant 1$ and $v^y + w^y > 1$, call procedure $P(w)$.
3. Add the items from the list of Add_v to a bin B_1 with the (actual) first item v from the list $(v_{m+1}, v_{m+2}, \ldots, v_n)$ of the remaining $BigX$-items if possible. If finally $1 \geqslant W^{x,y}(B_1) > \frac{1}{2}$, form a $BU(1)$ bunch. Otherwise, put the small items back into the list of non-assigned items, open a $BU(2)$

bunch, put v into the (empty) bin B_1 and call procedure $P(v)$.

The first lemma is obvious.

Lemma 1. *The bunches produced in Steps* 1, 2.1, 2.2 *and the* $BU(1)$ *bunches of Step* 3 *are all complete.*

Let BU_i be x-semicomplete (y-semicomplete). The next lemma shows that the total x-weight (y-weight) of the items in bunches $BU_1,\dots,BU_i$ is at least as big as the total x-weight (y-weight) of the items assigned to the bins in the optimal solution containing big items $w_1,\dots,w_i$ ($v_1,\dots,v_i$).

Lemma 2. *Let* BU_i *be a* x*-semicomplete bunch as produced by procedure* $P(w_i)$ *and let* $B_1^*,\dots,B_i^*$ *the bins in the optimal solution containing items* $w_1,\dots,w_i$. *Then*

$$W^x(BU_1)+\cdots+W^x(BU_i) \geqslant W^x(B_1^*)+\cdots+W^x(B_i^*).$$

Let BU_j *be a* y*-semicomplete bunch as produced by procedure* $P(v_j)$ *and let* $B_1^*,\dots,B_j^*$ *the bins in the optimal solution containing items* $v_1,\dots,v_j$. *Then*

$$W^y(BU_1)+\cdots+W^y(BU_j) \geqslant W^y(B_1^*)+\cdots+W^y(B_j^*).$$

Proof. BU_i is x-semicomplete means that Add_w_i is empty immediately after creating bunches $BU_1,\dots,BU_i$. Since $w_1^y \geqslant w_2^y \geqslant \cdots \geqslant w_i^y$, we have $Add\_w_1 \subseteq Add\_w_2 \subseteq \cdots \subseteq Add\_w_i$. That means all small items which are in bins $B_1^*,\dots,B_i^*$ have been assigned to $BU_1,\dots,BU_i$. Moreover, because of $v_1^x \geqslant v_2^x \geqslant \cdots \geqslant v_n^x$, and because i bins contain at most i *BigX*-items, the total x-weight of *BigX*-items in bins $B_1^*,\dots,B_i^*$ is at most $v_1^x+\cdots+v_i^x$. Using the fact that $v_1,\dots,v_i$ and $w_1\dots,w_i$ are all elements of $BU_1,\dots,BU_i$, the proof of the first inequality is finished.

The proof of the second assertion is analogous. Note that in this case for $j>m$ the total y-weight of *BigY*-items in bins $B_1^*,\dots,B_j^*$ is at most $w_1^y+\cdots+w_m^y$. □

Let BU_i denote the bunch containing *BigX*-item v_i ($i=1,\dots,n$). Let r_x denote the largest index such that BU_{r_x} is x-semicomplete and r_y the largest index such that BU_{r_y} is y-semicomplete, respectively. If no such bunch exists, the index is set to zero. We will see later on that the only x-incomplete bunches are bunches BU_i with $i \leqslant r_x$ and that the only y-incomplete bunches are bunches BU_j with $j \leqslant r_y$.

The x-incomplete $BU(2)$ bunches BU_i with $i > r_x$ are called *type* 1 *bunches*. The y-incomplete $BU(2)$ bunches BU_j with $j > r_y$ are called *type* 2 *bunches*. The next lemma describes the structure of type 1 and type 2 bunches. Especially, it shows that there are no bunches which are both type 1 and type 2.

Lemma 3. (a) *Type* 1 *bunches* BU_i ($i \in \{r_x+1,\dots,m\}$) *created in Step 2.3 have the following properties*:

$$v_i^x + w_i^x \leqslant 1, v_i^y + w_i^y > 1, W^x(BU_i) > \tfrac{1}{2}, W^y(BU_i) > 1 + w_i^y > \tfrac{3}{2}, Add\_w_i(x \geqslant y) = \emptyset.$$

There is a small item s *which is assigned to bin* B_2 *with* $s^y > 1 - W^y(B_1)$ *and* $s^y > s^x$.

(b) *Type* 2 *bunches* BU_j ($j \in \{r_y+1,\dots,m\}$) *created in Step* 2.3 *have the following properties*:

$$v_j^x + w_j^x > 1, v_j^y + w_j^y \leqslant 1, W^x(BU_j) > 1 + v_j^x > \tfrac{3}{2}, W^y(BU_j) > \tfrac{1}{2}, Add\_v_j(y > x) = \emptyset.$$

There is a small item s *which is assigned to bin* B_1 *with* $s^x > 1 - W^x(B_2)$ *and* $s^x > s^y$.

(c) *There are no type* 1 *bunches* BU_i *with* $i \in \{m+1,\dots,n\}$. *The type* 2 *bunches* BU_j ($j \in \{m+1,\dots,n\}$) *created in Step* 3 *have the following properties*:

$$W^x(BU_j) > \tfrac{3}{2}, \quad Add\_v_j(y > x) = \emptyset.$$

If there are type 1 *bunches* BU_i ($i \leqslant m$), *we have additionally*

$$W^y(BU_j) > 1 - w_i^y.$$

There is a small item s *which is assigned to bin* B_1 *with* $s^x > 1 - W^x(B_2)$ *and* $s^x > s^y$.

Proof. We start with part (a). The first three properties are trivial: $v_i^x + w_i^x \leqslant 1$ holds because BU_i is x-incomplete. If also $v_i^y + w_i^y \leqslant 1$ would hold, we could have formed a $BU(1)$ bunch in Step 2.1. The x-weight of BU_i is greater than $\frac{1}{2}$ since the bunch contains the *BigX*-item v_i.

Consider the application of procedure $P(w_i)$ in Step 2.3. By the fact that BU_i is not y-semicomplete, we know that there is at least one small item $s \in Add\_w_i$ which does not fit in bin B_1 and is therefore put into bin B_2 (where it fits by definition). Since BU_i is x-incomplete, the total x-weight of s and B_1 at this time is not greater than 1. Therefore, $s^y + W^y(B_1) > 1$. With $w_i^y > \frac{1}{2}$ the property $W^y(BU_i) > s^y + W^y(B_1) + w_i^y > 1 + w_i^y > \frac{3}{2}$ follows.

Remember that the x-coordinate of all elements of $NotAdd\_w_i(x \geqslant y)$ is as least as big as their y-coordinate and remember that the elements of the list of Add_w_i are sorted in non-increasing order of the ratio of x-coordinate and y-coordinate. Consequently, $s^x + W^x(B_1) < 1$ can only hold if already *all* elements of $Add\_w_i(x \geqslant y)$ are assigned to bin B_1, i.e. $Add\_w_i(x \geqslant y) = \emptyset$ and $s^y > s^x$.

Part (b) can be proven analogously to part (a), so we continue with part (c). Bunches BU_i $(i > m)$ contain no $BigY$-item. There could only occur a type 1 bunch BU_i in Step 3 if in the moment when the first small item does not fit into B_2, the total x-weight of B_2 would be smaller than $1 - v_i^x$. But in that case Step 3 would form a complete $BU(1)$ bunch with v_i before. (The elements of B_2 could be assigned to $BU(1)$ until the total y-weight of $BU(1)$ is between $\frac{1}{2}$ and 1 without exceeding a total x-weight of 1.) Because of this contradiction no type 1 bunch BU_i with $i > m$ exists.

The first two properties of type 2 bunches can be shown in the same way as in part (a). Also like in part (a) we can show that BU_j contains a small item s with $s_x > s_y$ and $s^x > 1 - W^x(B_2)$. Let BU_i be an arbitrary type 1 bunch with $i \leqslant m$. From part (a) we know that $Add\_w_i(x \geqslant y) = \emptyset$. Thus, $s \in NotAdd\_w_i(x \geqslant y)$ and $W^y(BU_j) \geqslant s^y > 1 - w_i^y$. □

In Step 4 of the algorithm type 1 bunches and type 2 are combined to $BU(4)$ bunches. By Lemma 3 it can be easily seen that these $BU(4)$ bunches are complete. The remaining type 1 bunches (or type 2 bunches, respectively) form with two (initially empty) bins $BU(4)$ bunches. Items from $Small(x \geqslant y)$ (or $Small(x > y)$, respectively) are assigned to these two additional bins. This is done in Step 5.

4. Form $BU(4)$ bunches consisting of an arbitrary type 1 bunch and an arbitrary type 2 bunch until no type 1 bunches or no type 2 bunches are left.
5. Consider the case that only type 1 bunches are left. (The case that there are only type 2 bunches, is treated analogously.) Sort the elements of $Small(x \geqslant y)$ in a list in non-increasing order of the values of the x-coordinate, i.e. $s_1^x \geqslant s_2^x \geqslant \cdots$. While $Small(x \geqslant y) \neq \emptyset$ and type 1 bunches exist, form $BU(4)$ bunches consisting of bins B_1, B_2 of a type 1 bunch and two empty bins B_3 and B_4. Take the (actual) first items of the list of $Small(x \geqslant y)$ and try to assign them to B_3 and to B_4.

Lemma 4. *All $BU(4)$ bunches produced in Step 4 are complete.*

Proof. The result follows directly from Lemma 3. □

The next lemma is only formulated for the case that only type 1 bunches are left after Step 4. If only type 2 bunches are left, the formulation is completely symmetric.

Lemma 5. *Let only type 1 bunches be left after Step 4. Then, all produced $BU(4)$ bunches are y-complete. If $Small(x \geqslant y) \neq \emptyset$ after Step 5, all $BU(4)$ bunches produced in Step 5 are complete.*

Proof. For an arbitrary $BU(4)$ bunch produced in Step 5 assume that the bins B_1, B_2 originate from a type 1 bunch BU_i with $BigY$-item w_i in B_2.

Consider first the y-coordinate. By Lemma 3 there is a small item s which is assigned to bin B_2 with $s^y > 1 - W^y(B_1)$ and $W^y(B_2) \geqslant w_i^y + s^y$. For each item $\tilde{s}$ assigned to B_3 or B_4 (there is at least one!), we have that

$$\tilde{s}^y > 1 - w_i^y \geqslant 1 - W^y(B_2) + s^y$$

holds, since $Add\_w_i(x \geqslant y) = \emptyset$ and thus $\tilde{s} \in NotAdd\_w_i(x \geqslant y)$. This gives

$$W^y(B_1) + W^y(B_2) + W^y(B_3) > (1 - s^y) + W^y(B_2) + (1 - W^y(B_2) + s^y) \geqslant 2.$$

Now assume $Small(x \geqslant y) \neq \emptyset$ after Step 5 and consider the x-coordinate. Recall that from Lemma 3 we know that $Add\_w_i(x \geqslant y) = \emptyset$ and therefore all items assigned to B_3 and B_4 are from $NotAdd\_w_i(x \geqslant y)$. Procedure $P(w_i)$ tested for each item of $NotAdd\_w_i(x \geqslant y)$ whether it fits into bin B_1. Consequently, we have for each $\tilde{s}$ assigned to B_3 or B_4 that

$\tilde{s}^x > 1 - W^x(B_1)$. Set $W^x(B_1) := 1/2 + \varepsilon$ with $\varepsilon > 0$. Since $W^x(Small(x \geqslant y)) \neq \emptyset$ after the construction of this $BU(4)$ bunch, there are at least four such small items $\tilde{s}$ assigned to B_3 and B_4 all with $\frac{1}{2} \geqslant \tilde{s}^x > \frac{1}{2} - \varepsilon$.

If $\varepsilon \leqslant 1/6$, we get

$$W^x(B_1) + W^x(B_3) + W^x(B_4) > \frac{1}{2} + \varepsilon + 4\left(\frac{1}{2} - \varepsilon\right) \geqslant 2.$$

If $\varepsilon > \frac{1}{6}$, then $W^x(B_1) > \frac{2}{3}$. Assume for example $W^x(B_3) \leqslant \frac{2}{3}$. All items which did not fit in B_3 have x-weight greater than $\frac{1}{3}$. We sorted in Step 5 the elements of $Small(x \geqslant y)$ in non-increasing order of the x-coordinate. Thus, all items assigned to B_3 (there are at least two of them!) have also x-coordinate greater than $\frac{1}{3}$, a contradiction to $W^x(B_3) \leqslant \frac{2}{3}$. Consequently, $W^x(B_3) > \frac{2}{3}$ and $W^x(B_4) > \frac{2}{3}$. It follows that $W^x(B_1) + W^x(B_3) + W^x(B_4) > 2$. □

The final two steps of the algorithm describe how to assign small items to the bins when the big items have been already put into bunches.

6. While $W^x(Small(x \geqslant y)) > \frac{1}{2}$ and $W^y(Small(y > x)) > \frac{1}{2}$, open two bins B_1 and B_2. Assign items of $Small(x \geqslant y)$ into B_1 and $Small(y > x)$ into B_2 (if possible).
   - 6.1 If $W^{x,y}(B_1 \cup B_2) > 1$, form a $BU(2)$ bunch from B_1 and B_2.
   - 6.2 If $W^{x,y}(B_1 \cup B_2) \leqslant 1$, put all items of B_1 and B_2 into one bin and form a $BU(1)$ bunch.
   - 6.3 If $W^x(B_1 \cup B_2) \leqslant 1$ and $W^y(B_1 \cup B_2) > 1$, take small items from B_2 and put them into B_1 until $W^y(B_1) \geqslant \frac{1}{2}$. Form a $BU(1)$ bunch consisting of B_1 and return the remaining items from B_2 to the set of non-assigned small items. (The case $W^x(B_1 \cup B_2) > 1$ and $W^y(B_1 \cup B_2) \leqslant 1$ is analogous.)
7. Assign the items of $Small(x \geqslant y)$ and $Small(y > x)$in arbitrary order to bins by First Fit.

Lemma 6. *All bunches produced in Step* 6 *are complete.*

Proof. Step 6.1 is clear. Since all assigned items are small, $W^{x,y}(BU(1)) > \frac{1}{2}$ holds for the $BU(1)$ bunch of Step 6.2 and it is complete. Since $W^x(B_1 \cup B_2) \leqslant 1$ in Step 6.3, we can take items from B_2 and put them into B_1 until we have $1 \geqslant W^y(B_1) > \frac{1}{2}$ without exceeding the bin capacity in the x-coordinate. □

Now it is easy to show our main result.

Theorem 1. *Let C^H denote the number of bins used by our algorithm. Then we have $C^H \leqslant 2C^*$.*

Proof. Assume that only type 1 bunches are left after the execution of Step 4. (If only type 2 bunches are left, the proof runs completely analogously.) Consider the case that $W^x(Small(x \geqslant y)) > 1/2$ after Step 6.

We perform our estimation by the x-coordinate. From Lemma 2 we know that the total x-weight of the items assigned to bunches $BU_1, \ldots, BU_{r_x}$ is at least as big as the total x-weight of items assigned to bins $B_1^*, \ldots, B_{r_x}^*$ in the optimal solution. Thus, the total x-weight which has to be assigned to the other bins in the optimal solution is at least as big as the total x-weight W^x which is assigned to the other bins in the heuristic solution. Hence

$$C^* \geqslant r_x + \lceil W^x \rceil.$$

All bunches produced in Steps 1 to Step 6 (besides possibly bunches $BU_1, \ldots, BU_{r_x}$) are x-complete. Also in Step 7 at most one bin B is produced with $W^x(B) \leqslant \frac{1}{2}$. If such a bin exists, all items from $Small(y > x)$ left after Step 6 can be assigned to that bin. Otherwise, these items are put to at most one extra bin. (Their total y-weight is not bigger than $\frac{1}{2}$.) Altogether, the heuristic produces besides the bunches $BU_1, \ldots, BU_{r_x}$ a set of t bins with total x-weight greater than $t/2$ and at most one further bin with x-weight not greater than $\frac{1}{2}$. This gives $W^x > t/2$ and

$$\lceil W^x \rceil \geqslant \frac{t}{2} + \frac{1}{2}.$$

Summarizing

$$C^H \leqslant 2r_x + t + 1 \leqslant 2r_x + 2\lceil W^x \rceil.$$

The claim follows.

Now consider the case that $W^x(Small(x \geqslant y)) \leqslant \frac{1}{2}$ after Step 6. By Lemma 5 all $BU(4)$ bunches are y-complete. Also if $Small(x \geqslant y) = \emptyset$ after Step 5 the remaining type 1 bunches are all y-complete. Thus, as before we can see that all bunches produced in Steps 1–6 are y-complete. Also in Step 7 at most

one bin B is produced with $W^Y(B) \leqslant 1/2$. Consequently, an analogous proof as above can be given for $W^x(Small(x \geqslant y)) \leqslant 1/2$ by estimation of the y-coordinate. □

3. Conclusions

In this paper we presented an elementary algorithm with absolute performance guarantee 2 for 2-DVPP. It is an interesting open problem to find approximation algorithms with absolute worst-case performance ratio better than 2 or at least approximation algorithms with asymptotic worst-case performance ratio better than 2.

References

[1] A. Caprara, H. Kellerer, U. Pferschy, Approximation schemes for ordered vector packing problems, Proceedings of Approx 2001, Lecture Notes in Computer Science, Vol. 2129, Springer, Berlin, 2001, pp. 63–74.

[2] A. Caprara, P. Toth, Lower bounds and algorithms for the 2-dimensional vector packing problem, Discrete Appl. Math. 111 (2001) 231–262.

[3] C. Chekuri, S. Khanna, On multi-dimensional packing problems, in: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99), ACM Press, New York, 1999.

[4] W. Fernandez de la Vega, G.S. Luecker, Bin packing can be solved within $1 + \varepsilon$ in linear time, Combinatorica 1 (1981) 349–355.

[5] M.R. Garey, R.L. Graham, D.S. Johnson, A.C. Yao, Resource constrained scheduling as generalized bin packing, J. Combin. Theory Ser. A 21 (1976) 257–298.

[6] F.C.R. Spieksma, A branch-and-bound algorithm for the two-dimensional vector packing problem, Comput. Oper. Res. 21 (1994) 19–25.

[7] D. Vercruyssen, H. Muller, Simulation in production, Technical Report, University of Gent, Belgium, 1987.

[8] G.J. Woeginger, There is no asymptotic PTAS for two-dimensional vector packing, Inform. Process. Lett. 64 (1997) 293–297.