Multi-dimensional packing with conflicts

Leah Epstein[1], Asaf Levin[2], and Rob van Stee[3,⋆]

[1] Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il.
[2] Department of Statistics, The Hebrew University, Jerusalem 91905, Israel. levinas@mscc.huji.ac.il.
[3] Department of Computer Science, University of Karlsruhe, D-76128 Karlsruhe, Germany. vanstee@ira.uka.de.

Abstract. We study the multi-dimensional version of the bin packing problem with conflicts. We are given a set of squares $V = \{1, 2, \ldots, n\}$ with sides $s_1, s_2, \ldots, s_n \in [0, 1]$ and a conflict graph $G = (V, E)$. We seek to find a partition of the items into independent sets of G, where each independent set can be packed into a unit square bin, such that no two squares packed together in one bin overlap. The goal is to minimize the number of independent sets in the partition.
This problem generalizes the square packing problem (in which we have $E = \emptyset$) and the graph coloring problem (in which $s_i = 0$ for all $i = 1, 2, \ldots, n$). It is well known that coloring problems on general graphs are hard to approximate. Following previous work on the one-dimensional problem, we study the problem on specific graph classes, namely, bipartite graphs and perfect graphs.
We design a $2 + \varepsilon$-approximation for bipartite graphs, which is almost best possible (unless $P = NP$). For perfect graphs, we design a 3.2744-approximation.

1 Introduction

Two-dimensional packing of squares is a well-known problem, with applications in stock cutting and other fields. In the basic problem, the input consists of a set of squares of given sides. The goal is to pack the input into bins, which are unit squares. A packed item receives a location in the bin so that no pair of squares have an overlap. The goal is to minimize the number of used bins.

However, in computer related applications, items often represent processes. These processes may have conflicts due to efficiency, fault tolerance or security reasons. In such cases, the input set of items is accompanied with a conflict graph where each item corresponds to a vertex. A pair of items that cannot share a bin are represented by an edge in the conflict graph between the two corresponding vertices.

Formally, the problem is defined as follows. We are given a set of squares $V = \{1, 2, \ldots, n\}$ whose sides are denoted by $s_1, s_2 \ldots, s_n$ and satisfy $s_i \in [0, 1]$

⋆ Research supported by Alexander von Humboldt Foundation.

for all $1 \leq i \leq n$. We are also given a conflict graph $G = (V, E)$. A valid output is a partition of the items into independent sets of G, together with a packing of the squares of each set into a unit square bin. The packing of a bin is valid if no two squares that are packed together in this bin overlap. The goal is to find such a packing with a minimum number of independent sets.

This problem is a generalization of the square packing problem [1], where $E = \emptyset$, and of the graph coloring problem, where $s_i = 0$ for all $i = 1, 2, \ldots, n$. It is well known that coloring problems on general graphs are hard to approximate. Following previous work on the one-dimensional problem, we study the problem on specific graph classes, namely, bipartite graphs and perfect graphs.

For an algorithm $\mathcal{A}$, we denote its cost on an input I by $\mathcal{A}(I)$, and simply by $\mathcal{A}$, if I is clear from the context. An optimal algorithm that uses a minimum number of bins is denoted by OPT. We consider the (absolute) approximation ratio that is defined as follows. The (absolute) approximation ratio of $\mathcal{A}$ is the infimum $\mathcal{R}$ such that for any input I, $\mathcal{A}(I) \leq \mathcal{R} \cdot \text{OPT}(I)$. We restrict ourselves to algorithms that run in polynomial time. The asymptotic approximation ratio is defined as be $\limsup_{n\to\infty} \sup_I \left\{ \frac{\mathcal{A}(I)}{\text{OPT}(I)} | \text{OPT}(I) = n \right\}$.

The one dimensional problem (where items are one-dimensional rather than squares) was studied on these graph classes. Jansen and Öhring [12] introduced the problem and designed approximation algorithms which work in two phases. The first phase is a coloring phase, where the graph is colored using a minimum number of colors. In the second phase, each independent set (which corresponds to a color class) is packed using a bin packing algorithm. Using this method, they obtained a 2-approximation algorithm for bipartite graphs and a 2.7-approximation algorithm for perfect graphs.

In [6], improved algorithms were designed. It was shown that the approximation ratio of the algorithm of [12] for perfect graphs is actually approximately 2.691, and a 2.5-approximation algorithm was designed. The algorithm applies a matching phase in which some pairs of relatively large items are packed in dedicated bins, and applies the methods of [12] as above on the remaining subgraph. An improved 1.75-approximation for bipartite conflict graphs was achieved by applying the algorithm of [12] on inputs with large enough values of OPT, while finding better solutions for inputs with small values of OPT.

Several papers [12, 11, 6] contain further results for additional graph classes. The paper [12] considered a class of graphs, on which the PRECOLORING EXTENSION problem (see [10, 16, 17]) can be solved in polynomial time. In this problem a graph is to be colored using a minimum number of colors with the constraint that some vertices already have given colors (a different color to each such vertex). This class contains chordal graphs, interval graphs, forests, split graphs, complements of bipartite graphs, cographs, partial K-trees and complements of Meyniel graphs. For these graphs, they designed a 2.5-approximation algorithm which is based on solving the PRECOLORING EXTENSION problem on the graph (where the items of size larger than $\frac{1}{2}$ are pre-colored each with a different color). In [6] an improved $\frac{7}{3}$-approximation algorithm, which is based

on a pre-processing phase in which subsets of at most three items are packed into dedicated bins, was designed.

For all $\varepsilon > 0$, Jansen and Öhring [12] also presented a $(2+\varepsilon)$-approximation algorithm for one-dimensional packing with conflicts on cographs and partial K-trees. Jansen [11] showed an asymptotic fully polynomial time approximation scheme for the one-dimensional problem on d-inductive graphs, where d is a constant. A d-inductive graph has the property that the vertices can be assigned distinct numbers $1, \ldots, n$ such that each vertex is adjacent to at most d lower numbered vertices. This includes the cases of trees, grid graphs, planar graphs and graphs with constant tree-width. Additional papers [20, 18] studied the one-dimensional problem on graphs that are unions of cliques, but their results are inferior to work of Jansen and Öhring [12].

The inapproximability results known for the two-dimensional and one-dimensional packing problems are as follows. Since standard bin packing (two-dimensional packing of squares and one-dimensional packing, respectively), is a special case of the problems with conflicts, the same inapproximability results holds for them as well. This means that the one-dimensional problem cannot be approximated up to a factor smaller than $\frac{3}{2}$, unless $P = NP$, (due to a simple reduction from the PARTITION problem, see problem SP12 in [8]). Also, the two-dimensional problem cannot be approximated up to a factor smaller than 2, unless $P = NP$, since it was shown in [15] that given a set of squares, it is NP-hard to check whether these squares can be packed into one bin. These results hold for the graph classes we consider since an empty graph (i.e., a graph with an empty edge set) is both bipartite and perfect.

Square packing was studied in many variants. An algorithm of approximation 2 (best possible unless $P = NP$) was shown in [22]. Unlike coloring problems, bin packing is often studied with respect to the asymptotic approximation ratio. An asymptotic approximation scheme was given by Bansal et al. [1, 2, 5]. This was the last result after a sequence of improvements [4, 13, 3, 14, 21, 7].

Our results. In this paper we design approximation algorithms for bipartite graphs and perfect graphs. For bipartite graphs, we give an algorithm of approximation ratio $2+\varepsilon$ for any $\varepsilon > 0$. Note that unlike the one-dimensional case, this is almost best possible unless $P = NP$. The algorithm chooses the best solution out of several algorithms, which are designed for various values of OPT. For perfect graphs we design algorithms which have clever pre-processing phases. We analyze an algorithm which chooses the best solution out of the outputs of all the algorithms we design. This results in an algorithm of approximation ratio at most 3.2744.

2 Bipartite graphs

In this section, we present an algorithm and analysis for the case where the conflict graph is bipartite. This algorithm will use the well-known square packing algorithm NEXT FIT DECREASING (NFD) [19] and a natural variant of it, FIRST FIT DECREASING (FFD), as subroutines. We begin by giving some properties of

these two algorithms in Section 2.1. In Section 2.2, we introduce a new algorithm called SixEleven, which is a variation of FFD which packs items differently in one special, crucial case. This helps to get a better area guarantee in a bin packed with SixEleven. We then describe our main algorithm for the cases $\text{OPT} = 1$ and $\text{OPT} = 2$ (Section 2.3), $\text{OPT} = 3$ (Section 2.4), OPT is a constant $k > 3$ (Section 2.5) and finally the case where OPT is not constant (Section 2.6). Since the value of OPT is unknown to the algorithm, the algorithm needs to apply all these possibilities and among these that output a valid solution, choose the one with the smallest cost. We will therefore assume that OPT is known to the algorithm (but make sure that the number of different algorithms applied is constant).

2.1 NFD and FFD

NFD packs items in slices, which are rectangular regions of the bin of width 1 that are stacked on top of each other starting from the bottom of the bin. The height of a slice is defined as the side of the first item packed into it. Each item is packed immediately to the right of the previously packed item, or in the next slice in case it does not fit in the current slice. When a new slice does not fit in the current bin, a new bin is opened for it. FFD works the same, but tries to put each new item in each slice that has been opened so far (to the right of the last item in the slice) instead of only trying the last slice or a new one. Regarding NFD and FFD, we have the following results.

Lemma 1 (Meir & Moser [19]). *Let L be a list of squares with sides $x_1 \geq x_2 \geq \ldots$ Then L can be packed in a rectangle of height $a \geq x_1$ and width $b \geq x_1$ using* Next Fit Decreasing *if one of the following conditions is satisfied:*

- *the total area of items in L is at most $x_1^2 + (a - x_1)(b - x_1)$.*
- *the total area of items in L is at most $ab/2$.*

In the following, we will abuse notation and use x_i to denote both the ith item in the input and its side, i.e., the length of one of its sides.

Lemma 2 (van Stee [22]). *Consider a bin that is packed by NFD, and suppose the largest item in this bin has side at most 1/3. If after packing this bin, there are still unpacked items with side at most $\frac{1}{3}$ left, then the total area of the items in the bin is at least $9/16$.*

2.2 Algorithm SixEleven

Algorithm SixEleven is displayed in Figure 1. It has the following properties.

Lemma 3. *Consider a set of squares where the largest item has side strictly more than $1/3$. If the two largest items have total side (i.e., sum of sides) at most 1, and the largest item that remains unpacked has side at most $1/5$, then SixEleven packs at least a total area of $6/11$ in this bin, unless it runs out of items.*

Input: A list of squares $\{x_1, \ldots, x_n\}$, sorted in order of nonincreasing side
Output: A packing of the input or a prefix of it in a single bin.

1. If $x_1 + x_2 + x_3 > 1$, but $x_1 + x_2 + x_4 \leq 1$, pack the three largest items as shown on the right. Pack the area A using NFD starting from the fourth item, then continue in area B with NFD (considering this to be a single slice), and finally pack area C using NFD.
2. Else, use FFD.

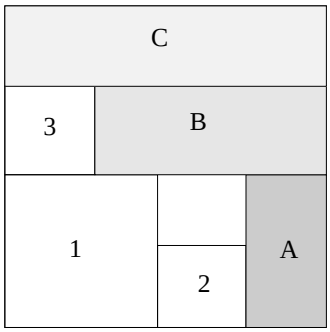

Fig. 1. Algorithm SixEleven

Define a *large item* to be an item with side more than 1/88. An item that is not large is said to be a *small item*. A large item is *huge* if its side is more than 1/3.

Definition 1. *A* good *set of squares is a set S with at least one of the following properties:*

1. *The two largest items in S have total side at most 1, and the total area of the large items is at most 6/11.*
2. *S contains only one large item.*

Theorem 1. *For any input set S which is good, SixEleven either packs S in one bin, or packs at least an area of 6/11 in the first bin.*

This Theorem implies that when SixEleven packs a good set, all the large items in the set are packed in the first bin.

2.3 The algorithm for OPT = 1 and OPT = 2

Recall that the conflict graph is bipartite. Thus, it is 2-colorable in all cases. If $\text{OPT} = 1$, we get that all items can be packed into a single bin, and therefore the conflict graph is empty. We can apply the 2-approximation from [22].

If $\text{OPT} = 2$, we act as follows. There are at most 18 huge items. Consider all partitions (a constant number) of the huge items into two sets L_1 and L_2. For the analysis it suffices to consider the iteration of the correct guess. So each such set of huge items can be packed with one bin (and we can find such a packing using the algorithm from Bansal et al. [1], which gives a constant time algorithm to pack a constant number of squares into a bin, is possible), and the coloring of the huge items (where the color of an item is determined by the set it is in) can be extended to a 2-coloring of the entire input as explained below.

For each connected component that contains a huge item the 2-coloring is defined uniquely (unless it contains at least two huge items and we get that

it is impossible to extend the coloring accordingly, in this case the partition of the huge items is incorrect), and it remains to decide on the 2-coloring of the connected components of the remaining items. For this problem we apply a similar idea to the one in [6] on the 1-dimensional case, only the partition into two sets must be done more carefully here. For each connected component we find its 2-coloring and we need to decide which color is red and which color is blue (in each of the connected components). We see the problem of balancing the area of blue items and red items as a load balancing problem. Let t be the number of connected components. For each connected component i, let c_i and d_i be the areas of items of the two colors in component i, we define $p_i = \max\{c_i, d_i\}$ and $\Delta(i) = \min\{c_i, d_i\}$. Clearly, each color has in total an area of at least $\sum_{i=1}^{t} \Delta(i)$. We define a load balancing problem on the residual area, i.e., we would like to balance the loads $p_i - \Delta(i)$ between two "machines", where assigning "job" i to machine 1 means that in component i, the color class of larger area got red color, and assigning "job" i to machine 2 means that in component i, the color class of larger area got blue color. Some "jobs" are pre-assigned to a machine if the coloring of this component is determined by the huge items. Therefore, we have a restricted assignment problem. This is a special case of load balancing on two unrelated machines, which admits an FPTAS, see [9].

Consider an optimal solution to the original bin packing problem. The total size of the items that are packed with L_i for $i = 1, 2$ is at most 1. Since we are using an FPTAS, where some area may be removed, the totals remain at most 1 and the total size of the larger set of items is at most 1.006 (for $\varepsilon = 0.006$).

Next we show that we can apply an algorithm based on SixEleven for each color class, which uses at most two bins (and four in total). First consider the case where the set of the huge items in this color has size at least 4/9. Then the huge items use at most one bin (using the packing of the algorithm from [1]), and for the other items, if by packing them using SixEleven, we need at least two bins, then we have an area guarantee of at least 9/16 in the second bin by Lemma 2, and this is a contradiction as $4/9 + 9/16 > 1.006$.

On the other hand, if the total area of the huge items is at most 4/9, then we use SixEleven on the complete color class. We would like to show that the area guarantee of the first packed bin is at least 4/9, if there is a second bin. If there is a single huge item and it has side at least 2/3 we are done. Otherwise, the huge item can fit next to any other item. If there are at least two huge items, since the huge items can fit into one bin, the sum of sides of the largest two items in at most 1. We get from the proof of Lemma 3 that if an item does not fit into the first bin, then the area guarantee is 6/11 in cases 1,2,4, no matter what the size of the next item is, and a guarantee of 1/2 in case 3, unless the bin contains exactly four items. Since the next item had side of at most 1/3, we get a guarantee of $\frac{4}{9}$ in this case (similarly to the proof for the case that this item is bounded by 1/5). So if there is a second bin, the first one has an area guarantee of $\frac{4}{9}$. If we are using three bins, then the second bin again has an area guarantee of 9/16 by Lemma 2, which again leads to a contradiction.

2.4 The algorithm for OPT = 3

We call items with side in $(1/3, 1/2]$ items of type 2, and larger items are type 1. In this section, items with side at most $1/88$ are called *small*, and the others are *large*. If $\text{OPT} = 3$, there are at most $3 \cdot 87^2$ large items. In constant time, find

- A two-coloring of these items that can be extended to a valid coloring for the entire input. This can be done by standard methods. We color the entire conflict graph ignoring the sizes of items.
- A packing of these items in at most three bins. This can be again done by checking all possible partitions of large items into three sets, and application of the algorithm of [1] on each set to pack it into a bin.

Note that the two results are unrelated and we do not require the packing to be consistent with the two-coloring. There are two cases. First, if the total area of the small items is at most $2 \cdot (\frac{87}{88})^2 \approx 1.9548$, do the following.

1. Use an arbitrary valid two-coloring for the small items.
2. Pack the largest set of small items in 2 bins, and the smallest set in at most 1 bin, using NFD.
3. Pack the large items in at most three bins according to the packing found above.

To see that Step 2 can indeed by applied, note that the smallest set has area at most $(\frac{87}{88})^2$, and the largest set has area at most twice this. The first bin packed for the largest set has area packed at least $(\frac{87}{88})^2$ by Lemma 1, leaving at most the same amount for the second bin, which can be packed there using NFD again by Lemma 1.

If the total area of the small items is more than $2 \cdot (\frac{87}{88})^2$, consider the packing for the large items (in at most 3 bins) that we have found. This packing gives us (at most) three sets, denoted by L_1, L_2, L_3. Each set may contain items of both colors. The total area of these items is at most 1.0452. In total, there are at most three items with side more than $1/2$, since all items can be packed in three bins.

We are going to *repack* these items so that each bin contains only items of one color. In this way we ensure that we do not pack conflicting items together. We next show the following auxiliary claim.

Claim. All large items can be packed in at most four bins. For any color, if not all items of that color are packed with large items, then the bins with large items have area guarantee of at least $6/11$.

We now have two cases. First, one of the colors (say blue) might be good. This means that if we pack all blue items using SixEleven, by Theorem 1 SixEleven packs an area of at least $6/11$ in the first blue bin (unless perhaps if it needs only one bin for *all* blue items). By Lemma 1, the area guarantee of any other bin for this color (except, always, the last one) is $(\frac{87}{88})^2 > 0.977$. This gives us the following area guarantees.

Bins needed for blue items	1	2	3	4
Total area guarantee of blue items packed	6/11	1.5228	2.5002	3
Maximum possible area of red items	3	2.4546	1.4772	0.4998
Packed in red bin 1, 2, 3	6/11	6/11	6/11	1/2
Packed in red bin 4, 5 (if needed)	0.977	0.977	-	-

The area guarantees for the red bins follow from Claim 2.4. Using this table, it is easy to verify that in this case (i.e., if the set of blue items is good) we never need more than six bins.

We give one example of such a verification. Suppose the total area of the blue items is 1.6, and the set of blue items is good. Then by the above table, we need at most three bins for the blue items. Since the total area guarantee for the first three red bins is $18/11 > 1.4 = 3 - 1.6$, we need at most three bins for the red items as well, so at most six bins in total.

If neither color is good, the large blue items are packed into *two* bins (either by SixEleven, or in some other way). In this case by Claim 2.4, we can pack the red items with area guarantees of 6/11 in the first two bins. Therefore, all large red items are packed in the first two bins since $12/11 > 1.0452$. Therefore, any further red bin that is packed using SixEleven (which uses FFD in this case) will again have an area guarantee of $(\frac{87}{88})^2 > 0.977$ by Lemma 1. Overall we find the following table.

Bins needed for blue items	1	2	3	4
Total area guarantee of blue items packed	6/11	12/11	2.068	3
Maximum possible area of red items	3	2.4546	1.909	0.932
Packed in red bin 1, 2	6/11	6/11	6/11	6/11
Packed in red bin 3, 4 (if needed)	0.977	0.977	0.977	-

Again, it can be verified that this is sufficient to pack all items in at most six bins in all cases.

2.5 The algorithm for OPT $= k > 3$

For any constant value k of OPT, we can find using Lemma 1 a value ε such that the area guarantee for NFD on items of side at most ε is at least $(k - 1.0452)/(k-1) = 1 - \frac{0.0452}{k-1}$. Then, if the small items have total area at most $k-1.0452$, we can pack them into at most k bins using NFD, and find an optimal packing for the items with side larger than ε using complete enumeration.

Else, the items with side at least ε have total area at most 1.0452. The proof of Claim 1 shows that *in case there are at most three items of type 1* we need at most four bins for all large items. We now show that we need at most $2k$ bins for all the items. If SixEleven needs more bins for both colors, this follows because the area guarantee in the four bins with large items is 24/11, so a total area of at most $k - \frac{24}{11}$ remains to be packed, and we have

$$k - \frac{24}{11} < (2k-6)\left(1 - \frac{0.0452}{k-1}\right) \qquad \text{for } k \geq 4. \tag{1}$$

So we need at most $2k-5$ bins for the small items of both colors: we lose (at most) one bin compared to (1) because there are two colors. (If there are less than four bins with large items, the area guarantee of the remaining bins improves.)

If SixEleven has already packed one color, then the small items of the other color have total area at most $\min(k, k-\frac{6}{11}(j-2))$ where $j \leq 4$ is the number of bins packed so far (there may be two almost empty bins that contain large items, since we have two colors). These items can be packed in at most $2k-j$ bins for $k \geq 4$, since

$$\min(k, k-\frac{6}{11}(j-2)) < (2k-j)\left(1-\frac{0.0452}{k-1}\right) \qquad \text{for } k \geq 4, j = 0, \ldots, 4. \quad (2)$$

The only case that is not covered yet is the case where there are **four** items of type 1 (since there cannot be more than four such items because the total size of items with side at least ε is at most 1.0452). If all these items are red (say), the blue items are good, and we pack the large red items in four bins. In case we need more bins for both colors, we now have five bins with area guarantee 6/11, and we can pack the remaining items in at most $2k-5$ bins since $k-\frac{30}{11} < (2k-6)(1-\frac{0.0452}{k-1})$ for $k \geq 4$. If one color is already packed, we can pack the remaining items into at most $2k-6$ bins by (1) if we packed five bins so far, and into at most $2k-j$ bins by (2) if we packed $j < 5$ bins so far.

If only one item of type 1 is blue, the blue items are still good. In this case the red items are also good if we exclude the two largest red items, so we need only four bins for all large items (again packing the red items as in Case 1A). Finally, if there are two blue items of type 1, we can pack the large items of each color into two bins, since removing the largest item of either color leaves a good set.

2.6 The algorithm for large OPT

Consider a fixed value $\varepsilon > 0$. There are two cases: if $\varepsilon \cdot \text{OPT} > 2$, color the items with two colors, and on each of them apply the APTAS of [1] for square packing. Since the minimum number of bins required to pack each color class is no larger than OPT, it needs only at most $2((1+\varepsilon)\text{OPT}+1) \leq (2+3\varepsilon)\text{OPT}$ bins. Else, $\text{OPT} \leq 2/\varepsilon$ which is a constant, so use the method from the previous section and use at most 2OPT bins. Note that for the case $\varepsilon \cdot \text{OPT} > 2$, we run just one algorithm, so in total we run at most $2/\varepsilon + 1$ polynomial-time algorithms and take the one that gives the best output.

3 An algorithm for perfect graphs

3.1 An algorithm for independent sets

Given an independent set of items, we use the following packing algorithm.
Algorithm Pack Independent Set (PackIS):

1. As long as there exists an item of side in $(\frac{1}{2}, 1]$, pack such an item in a bin.

2. As long as the number of items of side in $(\frac{1}{3}, \frac{1}{2}]$ is at least four, pack four such items in a bin.
3. As long as the number of items of side in $(\frac{1}{4}, \frac{1}{3}]$ is at least 9, pack 9 such items in a bin.
4. As long as the number of items of side in $(\frac{1}{5}, \frac{1}{4}]$ is at least 16, pack 16 such items in a bin.
5. If there are no items of side in $(\frac{1}{3}, \frac{1}{2}]$ left, pack the remaining items using NFD and halt.
6. Pack all items of side in $(0, \frac{1}{3}]$ using NFD. Call the resulting set of bins S, and let $m = |S|$. Let s_a be the side of the first item of bin m of S.
   Take bin m of S and remove all items from it. Pack its contents together with the remaining larger items (of side in $(\frac{1}{3}, \frac{1}{2}]$), possibly using a second bin, by applying algorithm SixEleven on the first bin, and NFD on the second bin. The items packed in the second adapted bin are those which did not fit into the first adapted bin.
   If a second bin is needed for the adapted packing and $s_a \leq \frac{1}{5}$, keep the first adapted bin packed with the items of side in $(\frac{1}{3}, \frac{1}{2}]$. Re-pack all other items (the ones in S plus the ones in the second adapted bin) once again with NFD. Note that this may affect the packing of bin $m-1$. Otherwise, the current packing (S without bin m together with one or two adapted bins) is given as output.

Note that there is at most one bin packed in the last step whose first packed item has side in the interval $(\frac{1}{k+1}, \frac{1}{k}]$, for $k = 2, 3, 4, 5$. To analyze our algorithm, we use three parameters, $\frac{8}{5} \leq r \leq \frac{16}{9}$, $\frac{1}{4} \leq \mu \leq \frac{2}{7}$ and $\frac{1}{9} \leq \nu \leq \frac{1}{7}$. These bounds imply

$$\nu \geq \frac{r}{16} \qquad \text{and} \qquad \mu \geq \frac{r}{9} \,. \tag{3}$$

We moreover require

$$r\frac{43}{99} + \mu \geq 1, \qquad r\frac{5}{16} + 4\nu \geq 1, \qquad r\frac{331}{648} + \nu \geq 1. \tag{4}$$

We assign weights as follows.

side	$(\frac{1}{2}, 1]$	$(\frac{1}{3}, \frac{1}{2}]$	$(\frac{1}{4}, \frac{1}{3}]$	$(0, \frac{1}{4}]$
weight	1	$\mu + r(x^2 - \frac{1}{9})$	$\nu + r(x^2 - \frac{1}{16})$	$r \cdot x^2$
expansion	1	$r + (\mu - \frac{r}{9})/x^2$	$r + (\nu - \frac{r}{16})/x^2$	r

Expansion is defined as the minimum ratio of weight over size of an item. By (3), it can be seen that the expansion of any item of side at most $\frac{1}{2}$ is at least r, so it is at least $\frac{8}{5}$.

Claim. Let ℓ be the number of bins created by Algorithm PackIS applied on a given color class. The sum of weights of items in this color class is at least $\ell - 1$.

3.2 The general algorithm

Algorithm Matching Preprocessing (PM):

1. Define the following auxiliary bipartite graph. One set of vertices consists of all items of side in $(\frac{1}{2}, 1]$. The other set of vertices consists of items of side in $(\frac{1}{4}, \frac{1}{2}]$. An edge (a, b) between vertices of items of sides $s_a > \frac{1}{2}$ and $s_b \leq \frac{1}{2}$ occurs if both following conditions hold.
   (a) $s_a + s_b \leq 1$.
   (b) $(a, b) \notin E(G)$.
   That is, if these two items can be placed in a bin together. If this edge occurs, we give it the cost μ if $s_b \geq \frac{1}{3}$ and ν otherwise.
2. Find a maximum cost matching in the bipartite graph.
3. Each pair of matched vertices is removed from G and packed into a bin together.
4. Let G' denote the induced subgraph over the items that were not packed in the preprocessing (i.e., during Steps 1,2,3).
5. Compute a feasible coloring of G' using $\chi(G')$ colors.
6. For each color class, apply the PackIS algorithm described above .

We analyze algorithm PM using weighting functions. Denote the weight function defined in the analysis of Algorithm PackIS for independent sets by w_1. We define the weight function for items packed into bins which are created in the preprocessing to be $1 - \mu$ for an item of side in $(\frac{1}{2}, 1]$ which is packed with an item of side in $(\frac{1}{3}, \frac{1}{2}]$, and $1 - \nu$ otherwise (i.e., if it is packed with an item of side in $\left(\frac{1}{4}, \frac{1}{3}\right]$).

We define a second weight function w_2 which is based on an optimal packing OPT of the entire input which we fix now. This weight function is defined differently from w_1 only for items of side in $(\frac{1}{2}, 1]$. Specifically, for a given such item x, consider the bin in which OPT packs x. If all items in this bin are of side in $(0, \frac{1}{4}]$, we define $w_2(x) = 1$. If the bin contains at least one other item of side larger than $\frac{1}{3}$, we define $w_2(x) = 1 - \mu$ and otherwise $w_2(x) = 1 - \nu$. Note that matching each item of side in $(\frac{1}{2}, 1]$, which got a weight strictly smaller than 1 with respect to w_2, with the largest item that shares its bin in OPT, gives a valid matching in the auxiliary bipartite graph. Therefore, if W_i denotes the total weight of all items with respect to the weight function w_i, then we have $W_1 \leq W_2$.

Theorem 2. *The approximation ratio of PM is at most 3.277344.*

Running an alternative algorithm which combines five possible preprocessing steps instead of just one improves the upper bound on the approximation ratio to 3.2743938. Details for this are omitted due to space constraints.

References

1. N. Bansal, J. Correa, C. Kenyon, and M. Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31(1):31–49, 2006.

2. N. Bansal and M. Sviridenko. New approximability and inapproximability results for 2-dimensional packing. In *Proceedings of the 15th Annual Symposium on Discrete Algorithms*, pages 189–196. ACM/SIAM, 2004.
3. A. Caprara. Packing 2-dimensional bins in harmony. In *Proc. 43rd Annual Symposium on Foundations of Computer Science*, pages 490–499, 2002.
4. F. R. K. Chung, M. R. Garey, and D. S. Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic and Discrete Methods*, 3:66–76, 1982.
5. J. Correa and C. Kenyon. Approximation schemes for multidimensional packing. In *Proceedings of the 15th ACM/SIAM Symposium on Discrete Algorithms*, pages 179–188. ACM/SIAM, 2004.
6. L. Epstein and A. Levin. On bin packing with conflicts. In *Proc. of the 4th Workshop on Approximation and online Algorithms (WAOA2006)*, pages 160–173, 2006.
7. L. Epstein and R. van Stee. Optimal online bounded space multidimensional packing. In *Proc. of 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 207–216, 2004.
8. M. R. Garey and D. S. Johnson. *Computers and intractability.* W. H. Freeman and Company, New York, 1979.
9. E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2):317–327, 1976.
10. M. Hujter and Z. Tuza. Precoloring extension, III: Classes of perfect graphs. *Combinatorics, Probability and Computing*, 5:35–56, 1996.
11. K. Jansen. An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization*, 3(4):363–377, 1999.
12. K. Jansen and S. Öhring. Approximation algorithms for time constrained scheduling. *Information and Computation*, 132:85–108, 1997.
13. C. Kenyon and E. Rémila. A near optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000.
14. Y. Kohayakawa, F. K. Miyazawa, Prabhakar Raghavan, and Yoshiko Wakabayashi. Multidimensional cube packing. *Algorithmica*, 40(3):173–187, 2004.
15. J. Y.-T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin. Packing squares into a square. *Journal on Parallel and Distributed Computing*, 10:271–275, 1990.
16. D. Marx. Precoloring extension. *http://www.cs.bme.hu/ dmarx/prext.html.*
17. D. Marx. Precoloring extension on chordal graphs. In *Graph Theory in Paris. Proceedings of a Conference in Memory of Claude Berge*, Trends in Mathematics, pages 255–270. Birkhäuser, 2007.
18. B. McCloskey and A. Shankar. Approaches to bin packing with clique-graph conflicts. Technical Report UCB/CSD-05-1378, EECS Department, University of California, Berkeley, 2005.
19. A. Meir and L. Moser. On packing of squares and cubes. *J. Combinatorial Theory Ser. A*, 5:126–134, 1968.
20. Y. Oh and S. H. Son. On a constrained bin-packing problem. Technical Report CS-95-14, Department of Computer Science, University of Virginia, 1995.
21. S. S. Seiden and R. van Stee. New bounds for multi-dimensional packing. *Algorithmica*, 36(3):261–293, 2003.
22. R. van Stee. An approximation algorithm for square packing. *Operations Research Letters*, 32(6):535–539, 2004.