Multi-dimensional packing with conflicts

Leah Epstein* Asaf Levin† Rob van Stee‡

August 8, 2007

Abstract

We study the two-dimensional version of the bin packing problem with conflicts. We are given a set of (two-dimensional) 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.

Topic: Algorithms and data structures

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 (two-dimensional) squares of given sides. The goal is to pack the input into bins, which are unit (two-dimensional) 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 (two-dimensional) squares $V = \{1, 2, \ldots, n\}$ whose sides are denoted by $s_1, s_2 \ldots, s_n$ and satisfy $s_i \in [0, 1]$ 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.

*Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il.

†Department of Statistics, The Hebrew University, Jerusalem 91905, Israel. levinas@mscc.huji.ac.il.

‡Department of Computer Science, University of Karlsruhe, D-76128 Karlsruhe, Germany. vanstee@ira.uka.de. Research supported by the Alexander von Humboldt Foundation.

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 $\limsup_{n\to\infty} \sup_I \left\{ \frac{\mathcal{A}(I)}{\text{OPT}(I)} | \text{OPT}(I) = n \right\}$. The absolute approximation ratio is useful when no inputs can be neglected. The asymptotic approximation ratio is used for problems where only the behavior of the algorithm for large enough inputs is of interest. Bin packing algorithms are typically measured using the asymptotic approximation ratio. However, coloring algorithms are usually measured using the absolute approximation ratio. Following previous work on (one-dimensional) packing with conflicts, we address the absolute approximation ratio in this paper.

One-dimensional packing without conflicts . The one dimensional problem (where both items and bins are one-dimensional rather than squares) was introduced in the early 70's [28, 7, 5]. Many variants of this problem has been studied ever since.

One-dimensional packing with conflicts . The one dimensional problem was studied on several graph classes, including perfect graphs and bipartite graphs. Jansen and Öhring [16] 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 [8], improved algorithms were designed. It was shown that the approximation ratio of the algorithm of [16] 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 [16] as above on the remaining subgraph. An improved 1.75-approximation for bipartite conflict graphs was achieved by applying the algorithm of [16] on inputs with large enough values of OPT, while finding better solutions for inputs with small values of OPT.

Several papers [16, 15, 8] contain further results for additional graph classes. The paper [16] considered a class of graphs, on which the PRECOLORING EXTENSION problem, where every precolored vertex is assigned a different color (see [13, 21, 22]), 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, mentioned above, on the graph (where the items of size larger than $\frac{1}{2}$ are precolored each with a different color). In [8] 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 [16] also presented a $(2 + \varepsilon)$-approximation algorithm for one-dimensional packing with conflicts on cographs and partial K-trees. Jansen [15] showed an asymptotic fully polynomial time approximation scheme for the one-dimensional problem on d-inductive (also called d-degenerate) 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 treewidth. Additional

papers [25, 23] studied the one-dimensional problem on graphs that are unions of cliques, but their results are inferior to work of Jansen and Öhring [16].

Hardness of approximability for packing without conflicts. 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 [10]). Also, the two-dimensional problem cannot be approximated up to a factor smaller than 2, unless $P = NP$, since it was shown in [20] 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 without conflicts. Square packing was studied in many variants. An algorithm of approximation 2 (best possible unless $P = NP$) was shown in [29]. 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, 6]. This was the last result after a sequence of improvements [4, 17, 3, 18, 27, 9].

Our results. In this paper we design the first approximation algorithms for bipartite graphs and perfect graphs. Prior to this work, no approximation algorithms for square packing with conflicts on any conflict graph were known. 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.

Our main result is for perfect conflict graphs, for which we design algorithms with 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. The only property of perfect graphs used by our algorithm is the existence of a polynomial time algorithm which finds a valid coloring of the graph using a minimum number of colors. An algorithm that finds such a coloring for perfect graphs is implied using the ellipsoid algorithm [11] (see also chapter 67 in [26]). Our algorithm is valid not only for perfect conflict graphs, but for any class of conflict graphs for which a minimum coloring can be found in polynomial time.

2 Bipartite graphs

In this section, we present an algorithm and analysis for the case where the conflict graph is bipartite, and establish the following theorem.

Theorem 1 *For every $\varepsilon > 0$, there exists a polynomial time approximation algorithm for square packing with conflicts, where the conflict graph is bipartite, with approximation ratio of at most $2 + \varepsilon$.*

The algorithm will use the well-known square packing algorithm NEXT FIT DECREASING (NFD) [24] 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 [24]) *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 [29]) *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 $9/16$.*

A *hereditary* condition on an input I is a condition which still holds if we remove some items from I. In particular, the conditions in Lemmas 1 and 2 are all hereditary. An *area guarantee ρ for algorithm $\mathcal{A}$* means that if we apply $\mathcal{A}$ on an input I and $\mathcal{A}(I)$ needs at least two bins then it uses at least one bin to pack items whose total area is at least ρ. The following technical lemma helps in the analysis of SixEleven.

Lemma 3 *Suppose we are given an area guarantee for NFD on input I and rectangle R that depends only on hereditary conditions, as long as not all items are packed in R. Then this area guarantee also holds for FFD.*

Proof Consider an input $I = \{x_1, \ldots, x_n\}$ and suppose that NFD as well as FFD do not pack I in one bin. Denote by $I' \subseteq I$ the subset which is packed in the first bin by FFD. We create a new input $I'' \subseteq I'$ from I' as follows. Remove from I' any item that is placed in an old slice by FFD, that is, not the most recently started slice. Denote the last (smallest) item in I'' by x_i, then $i < n$. Finally, add an item of side x_{i+1} temporarily to I''.

Consider the output of NFD for $I'' \cup x_{i+1}$. Since NFD never tries to use old slices, it can be seen by induction that each item is allocated to exactly the same slice and position as it was allocated by FFD on the input I'.

Regarding the item x_{i+1}, there are two options for FFD, since FFD did not place this item in the last slice.

Input: A list of squares of sides $\{x_1, \ldots, x_n\}$, 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 in Figure 2. 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.

Figure 1: Algorithm SixEleven

1. FFD puts x_{i+1} in an earlier slice, or

2. FFD does not put x_{i+1} in the first bin at all

In the first case, either NFD already "fails" before item x_{i+1} (NFD does not pack x_i in this bin), or NFD tries to put x_{i+1} into the last slice. However, if it were possible to pack x_{i+1} there, FFD would pack at least one item after x_i in the last slice, contradicting the definition of i.

In the second case, clearly NFD also does not put x_{i+1} in the first bin (since the packing so far is equal to the packing of FFD for the items in I'', and FFD tries more options to pack x_{i+1} than NFD does).

Thus in both cases NFD packs at most I'' without x_{i+1} in a bin, and $I'' \subseteq I'$. We have an area guarantee for I'', which then clearly also holds for the superset I' packed by FFD. □

2.2 Algorithm SixEleven

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

Lemma 4 *Consider a set of squares of sides $x_1 \geq x_2 \geq \ldots$ that is packed using SixEleven. Assume that at least one item remains unpacked and let x_τ be the side of the first such item. If the three conditions, $x_1 > 1/3$, $x_1 + x_2 \leq 1$ and $x_\tau \leq 1/5$ hold, then SixEleven packs at least a total area of $6/11$ in this bin.*

Proof We assume that there is at least one unpacked item. There are four cases. In Cases 1, 3 and 4, FFD is applied to the input; only in Case 2, Step 1 of SixEleven is applied. In Cases 1, 3 and 4, we will sometimes prove area guarantees for NFD (instead of FFD). This is sufficient by Lemma 3. Throughout this proof, we denote the side of the largest item by x, the side of the largest item in the second slice by y and the side of the largest unpacked item by α. Note that $y \geq \alpha$. Additionally, we also denote the sides of the items by $x_1, x_2, \ldots$ in nonincreasing order (so $x_1 = x$).

Case 1 *The largest three items have total side at most 1. I.e., $x_1 + x_2 + x_3 \leq 1$.*

In this case, NFD packs a total area of at least $x^2 + 2y^2$ in the first slice, and by Lemma 1, at least $y^2 + (1-y)(1-x-y) - \alpha^2 \geq (1-y)(1-x-y)$ in all other slices (since the height of the remaining part is $1-x$). Note that since the three largest items already have total side at most 1, NFD indeed allocates more than one slice. In fact at least three slices are opened, since the total height of these slices would be the sum of three items in the sequence, and even for the three largest items in the sequence x_1, x_2, x_3, we have $x_1 + x_2 + x_3 \leq 1$, so the second and third slices are indeed non-empty. In total, NFD packs at least an area of $A = x^2 + 2y^2 + (1-y)(1-x-y)$ in the bin. This expression has a global minimum of $6/11$ at $x = 4/11, y = 3/11$. In the remaining cases we assume that $x_1 + x_2 + x_3 > 1$.

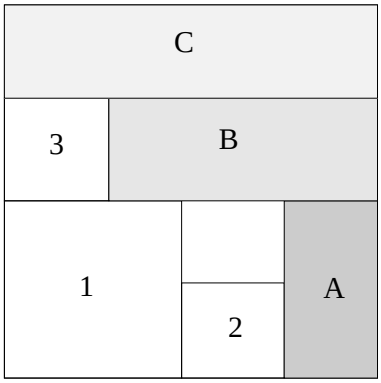

Figure 2: Alternative packing

Case 2 *The first, second and fourth item have total side at most 1. I.e.,* $x_1 + x_2 + x_4 \leq 1$.

In this case SixEleven uses the packing shown in Figure 2. We next argue that the rectangles A and C contain at least one item. For A, this holds because $x_1 + x_2 + x_4 \leq 1$. For C because the first item placed in it has side at most x_4, and $x_1 + x_3 + x_4 \leq x_1 + x_2 + x_4 \leq 1$.

B contains at least two items. The first item that SixEleven tries to pack there has side at most $x_4 \leq x_3$, where x_3 is the height of B, so at least one item fits height-wise. Moreover, $x_3 + 2x_4 \leq x_1 + x_2 + x_4 \leq 1$, so at least two items fit next to each other in B.

Denote the number of items in B by $k \geq 2$, and their sides by $z_1, \ldots, z_k$. In A, by Lemma 1, we pack items with total area at least $\frac{x(1-x-x_2)}{2} - z_1^2$. In B, we pack items with total area at least $\sum_{i=1}^{k} z_i^2$. In C, by Lemma 1, we pack items with total area at least $\frac{1-x-y}{2} - z_k^2$. In total, we pack total area of at least

$$\begin{aligned} P &= x^2 + x_2^2 + y^2 + \frac{x(1-x-x_2)}{2} + \sum_{i=2}^{k-1} z_i^2 + \frac{1-x-y}{2} \\ &= \frac{1}{2} + \frac{x^2}{2} - \frac{xx_2}{2} + x_2^2 + y^2 - \frac{y}{2} + \sum_{i=2}^{k-1} z_i^2. \end{aligned}$$

The condition are $x \geq x_2 \geq y \geq z_1 \geq \ldots \geq z_k$, $x + x_2 + y > 1$, and $\sum_{i=1}^{k} z_i > 1 - y - z_k$. From this expression, we simply omit the term $\sum_{i=2}^{k-1} z_i^2$. On the domain, $x > 1 - x_2 - y$, so $\frac{x^2 - xx_2}{2} > \frac{(1-x_2-y)(1-2x_2-y)}{2}$, and therefore P dominates

$$P_1 = 1 - \frac{3}{2}\left(y + x_2 - y^2 - yx_2\right) + 2x_2^2 = 1 - \frac{3}{2}(y + x_2)(1 - y) + 2x_2^2 \,.$$

We have $\partial P_1/\partial y = 0 \Leftrightarrow y = \frac{1}{2} - \frac{x_2}{2}$ and $\partial P_1/\partial x_2 = 0 \Leftrightarrow y = 1 - \frac{8}{3}x_2$. The global minimum of P_1 is attained at the point $x_2 = \frac{3}{13}, y = \frac{5}{13}$ where the constraint $y \leq x_2$ is not satisfied. Therefore, since P_1 is convex, the constraint $y \leq x_2$ is a binding constraint in the constrained optimum, and we can search for the minimum of P_1 with $y = x_2$. The minimum of $1 - \frac{3}{2}(2y)(1 - y) + 2y^2 = 5y^2 - 3y + 1$ is attained at $y = \frac{3}{10}$ and at the optimum P_1 has the value $\frac{11}{20} > \frac{6}{11}$.

Case 3 *NFD creates only two slices.* In this case and the next case, NFD packs exactly two items in the first slice: not more than two because we are not in Case 1, and at least two because the two largest items have total side at most 1. We make a case division based on the number of items packed in the second slice.

If the second slice contains only two items, the total side of these items is at least $4/5$ since the largest unpacked item has side at most $1/5$, and this then also holds for the total side of the first (largest) two items. By the arithmetic mean-geometric mean inequality, the total area of each pair of items is at least $2 \cdot (\frac{2}{5})^2$, so the total area overall is at least $4 \cdot (\frac{2}{5})^2 = \frac{16}{25} = 0.64 > \frac{6}{11}$.

Suppose the second slice contains three items. Denote their sides by y, z_1, z_2 in nonincreasing order. NFD packs a total area of at least $A = x^2 + 2y^2 + z_1^2 + z_2^2$, where $x + y > 4/5$ since there are only two slices, $y + z_1 + z_2 > 4/5$ and $x \geq y \geq z_1 \geq z_2$. Consider the vector $(\tilde{x}, \tilde{y}, \tilde{z}_1, \tilde{z}_2)$ which minimizes A on this domain. We have $\tilde{z}_1 = \tilde{z}_2$. This is so because suppose otherwise that $\tilde{z}_1 = \tilde{z}_2 + a$ for some $a > 0$, then the vector $(\tilde{x}, \tilde{y}, \tilde{z}_1 - a/2, \tilde{z}_2 + a/2)$ is also in the domain but the value of A is smaller, a contradiction. Assume therefore $z_1 = z_2$ and let $z = z_1 = z_2$.

Thus NFD packs a total area of at least $A_2 = x^2 + 2y^2 + 2z^2$ where $x > 4/5 - y$ and $z > (4/5 - y)/2$. Taking $x = 4/5 - y$ and $z = (4/5 - y)/2$, we find that A_2 dominates $A_3 = \frac{24}{25} - \frac{12}{5}y + \frac{7}{2}y^2$ on the domain. A_3 has a global minimum of $96/175 > 6/11$ for $y = 12/35$.

Suppose the second slice contains at least four items. NFD packs a total area of at least $x^2 + 2y^2 + \sum_{i=1}^{k} z_k^2$, where $y + \sum_{i=1}^{k} z_k > 1 - z_k$, $x + y > 1 - z_k$, and $k \geq 3$. We again find that in the minimum, $z_i = z$ for $i = 1, \ldots, k$, using a similar reasoning. Thus NFD packs a total area of at least $A = x^2 + 2y^2 + kz^2$ where $x \geq y \geq z$, $k \geq 3$, $y + kz > 1 - z$ and $x + y > 1 - z$. A is at least $A_2 = (1 - y - z)^2 + 2y^2 + kz^2$ on the domain since $x > 1 - y - z$. A_2 has a global minimum of $2k/(3k + 2)$ for $y = k/(3k + 2)$ and $z = 2/(3k + 2)$, which is monotonically increasing in k and is $6/11$ for $k = 3$.

Case 4 *NFD creates at least three slices.*

Denote the side of the fourth item by z (i.e., $z = x_4$). We find $x + x_2 + z > 1$ (since otherwise we are in Case 2), or $z > (1 - x - x_2)$. If NFD packs at least three items in the second slice, then by Lemma 1, NFD packs a total area of at least $x^2 + x_2^2 + y^2 + (1 - x - x_2)^2 + x_5^2 + (1 - x - y)/2 - \alpha^2 \geq x^2 + x_2^2 + y^2 + (1 - x - x_2)^2 + (1 - x - y)/2 = A$. This expression has a global minimum of $9/16 > 6/11$ for $x = \frac{1}{2}$, $x_2 = y = \frac{1}{4}$.

If NFD packs only two items in the second slice, there are two cases. If FFD packs some future item in one of the first two slices, we again find the area guarantee A, because the side of that item is larger than the side of the first unpacked item (just like $x_5 \geq \alpha$ above). Otherwise, since FFD packs at least three slices, FFD packs a total area of at least $A_3 = x^2 + 2y^2 + z^2 + 3(1 - z - y)^2$. This follows because the third slice has height less than 1/3, and therefore contains at least three items, and none of these items apparently fit in the second slice. Recall that $x_3 + x_4 + x_5 > 1$ and $y = x_3$. Therefore, $x > 1 - 2y$, and hence FFD packs a total area of at least $A_4 = (1 - 2y)^2 + 2y^2 + z^2 + 3(1 - z - y)^2$. A_4 has a global minimum of $17/27 > 6/11$ for $y = 11/27, z = 4/9$. □

Lemma 5 *Consider a set of squares. If the two largest items have total side 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.*

Proof If the side of the largest square is at most $1/3$, this follows from Lemma 2. Else, it follows from Lemma 4. □

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

Proof In case S contains only one large item, then if this large item has area at least 6/11, we are done. Else, the two largest items must have total side at most 1, so SixEleven packs at least four items in the first bin. Suppose SixEleven does not pack S in one bin, and it moreover packs a total area less than 6/11 in the first bin. Then the total area of the first four items in S is less than 6/11. If the first item in S has side at most $1/3$, SixEleven behaves like FFD, so by Lemma 2 the area packed is more than $6/11$, a contradiction. The largest item that remains unpacked must have side more than 1/5 by Lemma 4. Moreover, the cases 1,2, and 4 in the proof of that Lemma do not use the assumption that the largest unpacked item has side at most 1/5. (In particular, this covers the case where S contains only one large item of area less than $6/11$.) This also holds for the last subcase of Case 3 (at least four items in the second slice). Hence it must be the case that NFD creates two slices, where the first slice has two items and the second slice has two or three items.

Suppose the second slice has two items. Then the total side of the third, fourth and fifth item in S is more than 1, and their average area is more than $(\frac{1}{3})^2$. But then the area of each of the two largest items must both also be more than $(\frac{1}{3})^2$, giving that the total area of items with side more than 1/5 is more than 5/9 (since the fifth item has side more than 1/5, so then the first four items also have side at least 1/5), so S is not good, a contradiction.

Suppose the second slice contains three items. Denote their sides by y, z_1, z_2 in nonincreasing order. Let $a > \frac{1}{5}$ be the side of the first unpacked item. If $\frac{1}{5} < a \leq \frac{9}{40}$, NFD packs at least $A = x^2 + 2y^2 + z_1^2 + z_2^2$, where $x + y > 1 - a$ (since there is no third slice), $y + z_1 + z_2 > 1 - a$ (since a does not fit into the second slice) and $x \geq y \geq z_1 \geq z_2$. We conclude $x^2 + y^2 > \frac{1}{2}(1-a)^2$ and $y^2 + z_1^2 + z_2^2 > \frac{1}{3}(1-a)^2$ by the arithmetic mean-geometric mean inequality. But then we have that $A + a^2 \geq \frac{5}{6}(1-a)^2 + a^2$ which is monotonically non-increasing in this interval and larger than $6/11$ at $a = \frac{9}{40}$. Otherwise, we can follow the proof of Case 3 for at least four items in the second slice, but use the case $k = 2$ in which we get that the contents of the bin have a total area of at least $1/2$. Adding the area of the first unpacked item from S gives at least $\frac{881}{1600} > \frac{6}{11}$. We get that in both cases, S is not good. □

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 [29].

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 [8] 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 [12].

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 compelte 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 4 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 1 *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.*

Let us now consider the following two tables of area guarantees: Table 1 and Table 2.

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	-	-

Table 1: The set of blue items is good: SixEleven packs all large blue items in one bin

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	-

Table 2: the blue large items are placed in two bins

The first table concerns the case where one of the colors (called blue in the table) is *good*. This means that if we pack all blue items using SixEleven, by Theorem 2 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 $(\frac{87}{88})^2 > 0.977$. Furthermore, Claim 1 shows the printed area guarantees for the other color (red). 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 Table 1, 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.

The second table concerns the case where the large blue items are packed into *two* bins (either by SixEleven, or in some other way). In this case by Claim 1, 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. 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

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

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 finally 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 $k = 2, 3, 4, 5$.

As can be seen, some of the steps of this algorithm are based on a harmonic partition according to sides of items. The first to use such a partition in the design of bin packing algorithms were Lee and Lee [19].

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 2 *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 $\ell - 1$.*

Proof Consider the bins created in steps 1-4. The weights of items of sides in $(\frac{1}{2}, 1]$, $(\frac{1}{3}, \frac{1}{2}]$, $(\frac{1}{4}, \frac{1}{3}]$ are at least 1, $\frac{1}{4}$, $\frac{1}{9}$ respectively. The weight of an item of sides in $(\frac{1}{5}, \frac{1}{4}]$ is at least $\frac{r}{25} \geq \frac{8}{125} > \frac{1}{16}$. We get that the total weight of items in each one of these bins is at least 1. Next, consider bins created in steps 5 and 6. If there is at most one such bin we are done, therefore assume that at least two bins are created. In the execution of NFD, there is at most one bin whose first item has side in $(\frac{1}{k+1}, \frac{1}{k}]$, for $k = 3, 4$. Call these bins β and γ, and all other bins packed by NFD δ-bins. Note that bin β is the first bin packed in Step 5 or 6 (if it exists) and bin γ is the first or second bin packed in Step 5 or 6 (again, if it exists).

We first consider the case that SixEleven does not manage to pack all items in one bin in step 6. We distinguish the case where in step 6 $s_a \leq \frac{1}{5}$, and the case $s_a > \frac{1}{5}$.

Case 1 If $s_a \leq \frac{1}{5}$, then at least one δ-bin exists, so the last bin packed by NFD is a δ-bin. Consider first the bin β, if it exists, after NFD is run for the first time. Since β is the first bin packed in Step 5, the second round can only increase the area of items packed in this bin. Since this is not the last bin packed by NFD, it has a total packed area of at least $\frac{9}{16}$ [29]. Moreover, if there are at most three items in $(\frac{1}{4}, \frac{1}{3}]$, it can be deduced from [29] that the occupied area is actually at least $\frac{743}{1296}$. Note also that the expansion of all items in this bin is at least r.

The weight of an item of side s_b in $(\frac{1}{4}, \frac{1}{3}]$ is $\nu + r(s_b^2 - \frac{1}{16}) = rs_b^2 + \nu - \frac{r}{16}$. Thus if the bin contains s such items and its area guarantee is A, the total weight is at least $rA + s(\nu - \frac{r}{16})$. By (3), the minimum weight is achieved for minimal s. If $s \geq 4$, we get a weight of at least $r\frac{9}{16} + 4(\nu - \frac{r}{16}) = r\frac{5}{16} + 4\nu \geq 1$. Otherwise, since $s \geq 1$ we get a weight of at least $r\frac{743}{1296} + (\nu - \frac{r}{16}) = r\frac{331}{648} + \nu \geq 1$.

Consider the bin γ, if it exists, together with all δ-bins, and otherwise (i.e., if there is no bin γ) the δ-bins only. We consider these bins after the second round of NFD. Let $j > 1$ be the total amount of bins , and $y_1, \ldots, y_j$ the sides of the first items packed in these bins, where $\frac{1}{4} \leq y_1 \leq \ldots \leq y_j$.

By Lemma 1, each bin which is started by the item of side y_i for $i < j$ has an occupied area of at least $y_i^2 + (1 - y_i)^2 - y_{i+1}^2$. This gives a total of at least $y_1^2 + \sum_{i=1}^{j-1}(1 - y_i)^2$. Since $y_i \leq \frac{1}{5}$ for $i > 1$ we get at least $y_1^2 + (1 - y_1)^2 + (j - 2)\frac{16}{25}$. On the domain, this function is minimized for $y_1 = \frac{1}{4}$, and we get an area of at least $(j - 1)\frac{5}{8}$ and thus (since $r \geq \frac{8}{5}$) a total weight of at least $j - 1$.

Consider the bin packed using SixEleven. Since at least one item did not fit into it, and since this item has side no larger than $\frac{1}{5}$ (since already s_a belongs to this group of items, and further items can only be smaller), by Lemma 4 the area packed in this bin is at least $\frac{6}{11}$. Let t be the number of items of side in $(\frac{1}{3}, \frac{1}{2}]$ in this bin. The weight of an item of side s_c in $(\frac{1}{3}, \frac{1}{2}]$ is $\mu + r(s_c^2 - \frac{1}{9}) = rs_c^2 + \mu - \frac{r}{9}$, thus if the bin contains t such items and its area guarantee is B, the total weight is at least $rB + t(\mu - \frac{r}{9})$. By (3), the minimum weight is achieved for minimal t. We get a weight of at least $r\frac{6}{11} + (\mu - \frac{r}{9}) = \frac{43}{99}r + \mu \geq 1$.

Case 2 Consider now the case $s_a > \frac{1}{5}$. Note that in this case S contains at most two bins ($m \leq 2$), β and γ. Thus, the adapted bin was either β or γ. We consider both cases together, where the possible bin β can be analyzed as above. We analyze the two adapted bins together and show the total weight in them is at least 1. If the first item in the second bin is of side at most $\frac{1}{5}$, then by Lemma 4, the area packed in the first adapted bin is at least $\frac{6}{11}$. Otherwise, the only case of that theorem that requires this condition is if NFD is the algorithm which is used as a procedure by SixEleven, it creates two slices, and the second one has two or three items (see Case 3).

If there are two items in the second slice, the sum of the sides of every two items is more than $1 - s_a$. This gives a total area of at least $4(\frac{1-s_a}{2})^2 + s_a^2 = 2s_a^2 - 2s_a + 1$ for all five items, which has the minimum value $\frac{5}{9} > \frac{6}{11}$. Otherwise, the proof for three items in the second slice gives an area of at least $\frac{1}{2}$ (where the side of the item that does not fit into the bin may be arbitrary). In the case $\frac{1}{4} \leq s_a \leq \frac{1}{3}$, we have a total area of at least $\frac{9}{16} > \frac{6}{11}$ [29]. Otherwise, let s_z denote the side of the first item z in the first adapted bin. Let s_y be the side of the first item in the second slice and let t_1, t_2, t_3 be the sides of the two additional items in the second slice and the item which did not fit. We have a total area of at least $s_z^2 + 2s_y^2 + t_1^2 + t_2^2 + t_3^2$, where $s_z + s_y + t_3 > 1$ and $s_y + t_1 + t_2 + t_3 > 1$. This function is minimized for $t_1 = t_2 = t_3 = t$, $t = \frac{1-s_y}{3}$, $s_z = 1 - t - s_y = \frac{2(1-s_y)}{3}$, and achieves a minimal value for $s_y = 0.28$ which is $0.56 > \frac{6}{11}$.

As shown above, the total weight of a set of items of total area at least $\frac{6}{11}$, where at least one item has side in $(\frac{1}{3}, \frac{1}{2}]$, is at least 1.

Case 3 SixEleven does manage to pack all items in a single bin. In this case, the number of bins is the same as in the case where we would run only step 5 on items with side in $(0, \frac{1}{3}]$. Thus we can apply the analysis from Case 1 above for bins β, γ and the δ-bins and note that we now pack strictly more items (and therefore weight) in the same amount of bins. □

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 $W_1 \leq W_2$.

To prove an upper bound for PM, we first prove the following lemmas. Let ω_2 be an upper bound on the amount of weight according to w_2 that a set of items packed into a single bin can have.

Lemma 6 *Consider a partitioning of the input into sets, where each set is independent. Some of the sets consist of items that can be packed into a single bin, and have a total weight at least* 1 *according to* w_1*. Let* k *be the number of independent sets that do not follow this rule. These sets are packed using the algorithm PackIS. The number of packed bins is at most* $\omega_2\mathrm{OPT}+k$*.*

Proof Consider the k sets defined above, let ℓ_i be the number of bins resulting from set i, and ℓ the total number of bins including also sets that result in one bin of total weight at least 1. Using Claim 2 we find that the total weight of items in set i is at least ℓ_i-1. Since there are k such sets, the total weight according to w_1 is at least $\ell-k$, i.e. $W_2 \geq W_1 \geq \ell-k$. According to the definition of ω_2, we have $W_2 \leq \omega_2\mathrm{OPT}$ which proves the claim. □

Lemma 7 *The approximation ratio of PM is at most* ω_2+1*.*

Proof PM creates the independent sets using an optimal coloring algorithm. Therefore, $k \leq$ OPT, since k is the minimum number of colors required to color a subset of the input. □

Theorem 3 *The approximation ratio of PM, for square packing with conflicts, where the conflict graph is perfect, is at most 3.277344.*

Proof We need to analyze the total weight in packed bins of OPT. We first compute this value as a function of the parameters.

A bin with one item a of side $s_a \in (\frac{1}{2},1]$ and all other items no larger than $\frac{1}{4}$ has weight of at most

$$1+(1-s_a^2)r < \frac{3}{4}\cdot r+1.$$

Given a bin with one item a of side $s_a \in (\frac{1}{2}, 1]$ and all other items no larger than $\frac{1}{3}$, let s be the number of items of side in $(\frac{1}{4}, \frac{1}{3}]$. Clearly $s \leq 5$. The bin has a total weight of at most $1 - \nu + s\nu + (1 - \frac{1}{4} - \frac{s}{16})r = 1 + \frac{3}{4} \cdot r + (s-1)\nu - \frac{sr}{16}$. By (3), the expression is maximized for the largest value of s, giving

$$1 + \frac{7r}{16} + 4\nu. \tag{5}$$

Finally, consider a bin which consists of an item of side in $(\frac{1}{2}, 1]$ and at least one item of side in $(\frac{1}{3}, \frac{1}{2}]$. Let s and t be the number of items of sides in $(\frac{1}{4}, \frac{1}{3}]$ and $(\frac{1}{3}, \frac{1}{2}]$, respectively. Note that $s+t \leq 5$ and $1 \leq t \leq 3$. The bin has a total weight of at most $1-\mu+s\nu+t\mu+(1-\frac{1}{4}-\frac{t}{9}-\frac{s}{16})r < 1+0.75 \cdot r+s\nu+(t-1)\mu-\frac{tr}{9}-\frac{sr}{16}$. Since $\mu - \frac{r}{9} \geq 0$, the expression is maximized when $s+t$ is maximal, i.e. we need to consider the three cases $t = 1, s = 4$; $t = 2, s = 3$; $t = 3, s = 2$. We get the three bounds $1 + \frac{7r}{18} + 4\nu$, $1 + \frac{49r}{144} + 3\nu + \mu$, $1 + \frac{7r}{24} + 2\nu + 2\mu$. The first two are dominated by the last bound and/or by (5).

Given a bin where all items are of side no larger than $\frac{1}{2}$, let s and t be the number of items of sides in $(\frac{1}{4}, \frac{1}{3}]$ and $(\frac{1}{3}, \frac{1}{2}]$, respectively. Clearly $s + t \leq 9$ and $t \leq 4$. The bin has a total weight of at most $s\nu + t\mu + (1 - \frac{t}{9} - \frac{s}{16})r < r + s\nu + t\mu - \frac{tr}{9} - \frac{sr}{16}$. Again, the expression is maximized for maximal $s+t$. We need to consider the five cases $t = 0, s = 9$; $t = 1, s = 8$; $t = 2, s = 7$; $t = 3, s = 6$; $t = 4, s = 5$. This gives the five bounds $\frac{63r}{144} + 9\nu$, $\frac{56r}{144} + 8\nu + \mu$, $\frac{49r}{144} + 7\nu + 2\mu$, $\frac{42r}{144} + 6\nu + 3\mu$, $\frac{35r}{144} + 5\nu + 4\mu$. Obviously, only the first and the last need to be considered. Since $\nu \leq \frac{1}{7}$, the first is dominated by (5). Since $\mu \leq \frac{2}{7}$, the last one is dominated by $1 + \frac{7r}{24} + 2\nu + 2\mu$.

We are left with the following bound,

$$\max\{\frac{3r}{4} + 1, 1 + \frac{7r}{16} + 4\nu, 1 + \frac{7r}{24} + 2\nu + 2\mu\} .$$

Running a linear program we find that an upper bound on this value is approximately 2.277344, which is achieved for $r \approx 1.7031$, $\mu \approx 0.2603$, $\nu \approx 0.13$. This gives an upper bound of 3.277344 on the approximation ratio. □

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. The details of this algorithm are in the appendix.

4 Conclusion

In this paper we addressed the approximability of square packing with conflicts. Our study focuses on the absolute approximation ratio as is common for coloring problems. The upper bounds which we proved on the absolute approximation ratio of our algorithm clearly holds for the asymptotic approximation ratio as well. However, all the known approximability results, which are mentioned in the introduction, do not hold in this case. An interesting research direction would be to find whether an Asymptotic Polynomial Time Approximation Scheme (APTAS) exists for some square packing with conflicts, for some class of (non-empty) conflict graphs.

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] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation algorithms*. PWS Publishing Company, 1997.

[6] 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.

[7] J. Csirik and G. J. Woeginger. On-line packing and covering problems. In *A. Fiat and G. J. Woeginger, editors*, Online Algorithms: The State of the Art, pages 147–177, 1998.

[8] 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.

[9] 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.

[10] M. R. Garey and D. S. Johnson. *Computers and intractability*. W. H. Freeman and Company, New York, 1979.

[11] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1993.

[12] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2):317–327, 1976.

[13] M. Hujter and Z. Tuza. Precoloring extension, III: Classes of perfect graphs. *Combinatorics, Probability and Computing*, 5:35–56, 1996.

[14] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68–72, 1989.

[15] K. Jansen. An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization*, 3(4):363–377, 1999.

[16] K. Jansen and S. Öhring. Approximation algorithms for time constrained scheduling. *Information and Computation*, 132:85–108, 1997.

[17] 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.

[18] Y. Kohayakawa, F. K. Miyazawa, Prabhakar Raghavan, and Yoshiko Wakabayashi. Multidimensional cube packing. *Algorithmica*, 40(3):173–187, 2004.

[19] C. C. Lee and D. T. Lee. A simple online bin packing algorithm. *Journal of the ACM*, 32(3):562–572, 1985.

[20] 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.

[21] D. Marx. Precoloring extension. *http://www.cs.bme.hu/ dmarx/prext.html*.

[22] 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.

[23] 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.

[24] A. Meir and L. Moser. On packing of squares and cubes. *J. Combinatorial Theory Ser. A*, 5:126–134, 1968.

[25] 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.

[26] A. Schrijver. *Combinatorial optimization polyhedra and efficiency*. Springer-Verlag, 2003.

[27] S. S. Seiden and R. van Stee. New bounds for multi-dimensional packing. *Algorithmica*, 36(3):261–293, 2003.

[28] J. D. Ullman. The performance of a memory allocation algorithm. Technical Report 100, Princeton University, Princeton, NJ, 1971.

[29] R. van Stee. An approximation algorithm for square packing. *Operations Research Letters*, 32(6):535–539, 2004.

A Improved upper bound for perfect graphs

To achieve a better upper bound, we suggest four new preprocessing steps. Each possible preprocessing yields a different algorithm. Our main algorithm runs each one of the five algorithms, and it chooses the solution with the minimum number of bins.

The general structure of our algorithms is as follows. Perform a preprocessing which creates some packed bins. Then steps 4-6 of PM are applied. Each one of the four alternative pre-processing steps creates a collection of subsets, whose ground set is the set of items in the input. Each subset has at most k_i items ($2 \leq i \leq 5$). We are using an algorithm of Hurkens and Schrijver [14] for approximating the maximum (unweighted) set packing problem. Their algorithm finds a sub-collection of subsets, such that every pair of subsets is disjoint. The cardinality of the output sub-collection is at least a $\frac{2}{k_i} - \varepsilon$ fraction of the largest such sub-collection. The preprocessing packs each such subset into a separate bin.

The four collections are as follows.

2. Sets of four items that can fit into one bin, where one item has side in $(\frac{1}{2}, 1]$ and three items have sides in $(\frac{1}{3}, \frac{1}{2}]$. Therefore $k_2 = 4$.

3. Sets of three items that can fit into one bin, where one item has side in $(\frac{1}{2}, 1]$ and two items have sides in $(\frac{1}{3}, \frac{1}{2}]$. Therefore $k_3 = 3$.

4. Sets of six items that can fit into one bin, where one item has side in $(\frac{1}{2}, 1]$ and five items have sides in $(\frac{1}{4}, \frac{1}{2}]$. Therefore $k_4 = 6$.

5. Sets of five items that can fit into one bin, where one item has side in $(\frac{1}{2}, 1]$ and four items have sides in $(\frac{1}{4}, \frac{1}{2}]$. Therefore $k_5 = 5$.

We use $w_1[1] = w_1$ and $w_2[1] = w_2$. Furthermore we define four additional weight functions, $w_1[i]$ and $w_2[i]$ for $i = 2, 3, 4, 5$. The functions $w_2[i]$ are based on a fixed optimal packing as $w_2[1]$. The weight function for items of side in $(0, \frac{1}{2}]$ is as before. Items of side in $(\frac{1}{2}, 1]$ get weight 1 except for special cases as described below.

2. An item of side in $(\frac{1}{2}, 1]$ that in the optimal packing shares a bin with three items of side in $(\frac{1}{3}, \frac{1}{2}]$ gets weight $1 - \frac{3\mu(1-2\varepsilon)}{2}$ according to $w_2[2]$. Let N_2 be the number of such items.

3. An item of side in $(\frac{1}{2}, 1]$ that in the optimal packing shares a bin with at least two items of side in $(\frac{1}{3}, \frac{1}{2}]$ gets weight $1 - \frac{4\mu(1-\frac{3\varepsilon}{2})}{3}$ according to $w_2[3]$. Let N_3 be the number of such items.

4. An item of side in $(\frac{1}{2}, 1]$ that in the optimal packing shares a bin with five items of side in $(\frac{1}{4}, \frac{1}{2}]$ gets weight $1 - \frac{5\nu(1-3\varepsilon)}{3}$ according to $w_2[4]$. Let N_4 be the number of such items.

5. An item of side in $(\frac{1}{2}, 1]$ that in the optimal packing shares a bin with at least four items of side in $(\frac{1}{4}, \frac{1}{2}]$ gets weight $1 - \frac{8\nu(1-\frac{5\varepsilon}{2})}{5}$ according to $w_2[5]$. Let N_5 be the number of such items.

Next, we describe the functions $w_1[i]$. Similarly to $w_2[i]$, the only items that get special weights are the ones of side in $(\frac{1}{2}, 1]$.

2. An item of side in $(\frac{1}{2}, 1]$ that is packed into a bin during preprocessing gets weight $1 - 3\mu$ according to $w_1[2]$. Note that the number of such items is at least $(\frac{2}{4} - \varepsilon)N_2$.

3. An item of side in $(\frac{1}{2}, 1]$ that is packed into a bin during preprocessing gets weight $1 - 2\mu$ according to $w_1[3]$. Note that the number of such items is at least $(\frac{2}{3} - \varepsilon)N_3$.

4. An item of side in $(\frac{1}{2}, 1]$ that is packed into a bin during preprocessing gets weight $1 - 5\nu$ according to $w_1[4]$. Note that the number of such items is at least $(\frac{2}{6} - \varepsilon)N_4$.

5. An item of side in $(\frac{1}{2}, 1]$ that is packed into a bin during preprocessing gets weight $1 - 4\nu$ according to $w_1[5]$. Note that the number of such items is at least $(\frac{2}{5} - \varepsilon)N_5$.

Let W be the sum of regular weight of all items. Let $W_1[i]$ and $W_2[i]$ be the sums of weights of all items according to $w_1[i]$ and $w_2[i]$.

We have $W_2[2] = W - \frac{3}{2}\mu(1 - 2\varepsilon)N_2 = W - \frac{3\mu N_2}{2} + 3\mu\varepsilon N_2$, $W_2[3] = W - \frac{4}{3}\mu(1 - \varepsilon)N_3$, $W_2[4] = W - \frac{5}{3}\nu(1 - \varepsilon)N_4$, $W_2[5] = W - \frac{8}{5}\nu(1 - \varepsilon)N_5$.

On the other hand we have, $W_1[2] \leq W - 3\mu(\frac{1}{2} - \varepsilon)N_2 = W - \frac{3\mu N_2}{2} + 3\mu\varepsilon N_2 = W_2[2]$, $W_1[3] \leq W - 2\mu(\frac{2}{3} - \varepsilon)N_3 = W - \frac{4\mu N_3}{3} + 2\mu\varepsilon N_3 = W_2[3]$, $W_1[4] \leq W - 5\nu(\frac{1}{3} - \varepsilon)N_4 = W - \frac{5\nu N_4}{3} + 5\nu\varepsilon N_4 = W_2[4]$, $W_1[5] \leq W - 4\nu(\frac{2}{5} - \varepsilon)N_5 = W - \frac{8\nu N_5}{5} + 4\nu\varepsilon N_5 = W_2[5]$.

To prove an improved upper bound, let $\omega_2[i]$ be an upper bound on the amount of weight according to $w_2[i]$ that a set of items packed into a single bin can have. As in Lemma 6, for each weight function, we analyze the total amount of weight that can be packed into a single bin of OPT.

The analysis for the first pre-processing is the same as in the proof of Theorem 3. Since bins without an item of side in $(\frac{1}{2}, 1]$ have the same weight in all cases and no reductions, then we only need to consider the extreme cases as in the proof of Theorem 3.

There are 20 types of bins that need to be considered and we analyze each type according to every preprocessing. We denote items of side in $(\frac{1}{3}, \frac{1}{2}]$ by A and items of side in $(\frac{1}{4}, \frac{1}{3}]$ by B. Items of side in $(\frac{1}{2}, 1]$ are denoted by L.

Since we choose the solution with smallest number of bins, we use a convex combination instead. We use parameters $\alpha[i]$ (for $1 \leq i \leq 5$ such that $\sum_{i=1}^{5} \alpha[i] = 1$.

For each bin type j ($1 \leq j \leq 20$), we compute $t_2[i][j]$ which is the largest amount of weight that can be packed in a bin of type j according to weight function $w_2[i]$. We use the following lemma. Let $\omega_2' = \max_j \{\sum_{i=1}^{5} \alpha[i] t_2[i][j]\}$.

Lemma 8 *The approximation ratio of the algorithm which chooses the best out of the five solutions is at most* $\omega_2' + 1$.

Proof We define a new weight function $w_2' = \sum_{i=1}^{5} \alpha[i] \cdot w_2[i]$. For each i, the sum of the total weight according to $w_2[i]$ and the chromatic number, is at least the cost of the algorithm. Therefore, this holds also for w_2' and the cost of the best solution among the five algorithms. □

Using Matlab, we were able to find that using the values $r = 1.699191$, $\mu = 0.261967$, $\nu = 0.132049$, $\alpha[1] = 0.5872688$, $\alpha[2] = 0.120419$, $\alpha[3] = 0.052589$, approximately 3.2743938. We summarize with the following theorem.

Theorem 4 *The approximation ratio of the combined algorithm, for square packing with conflicts, where the conflict graph is perfect, is at most 3.274394.*