Online Square Packing With Rotation

Shahin Kamali[*] Pooya Nikbakht[†]

Abstract

In the square packing problem, the goal is to place a multi-set of square items of different side lengths in $(0, 1]$ into a minimum number of square bins of uniform side length 1. In the online setting, the multi-set of items forms a sequence that is revealed in an online and sequential manner. When an item is revealed, an online algorithm has to place it into a square bin without any prior knowledge of the forthcoming items. All existing results for the online square packing are restricted to the case when square items are placed orthogonally to the square bins. In this paper, we provide an algorithm with an asymptotic competitive ratio of 2.306 when squares are allowed to be rotated.

1 Introduction

An instance of the *square packing problem* is defined with a multiset of *squares-items* of different sizes in the range $(0, 1]$. The goal is to place these squares into a minimum number of unit *square bins* in a way that two squares-items placed in the same square bin do not intersect. The problem is a generalization of the classical bin packing problem into two dimensions. As such, we sometimes refer to the squares-items simply as *items* and square bins as *bins*. A square item can be recognized by the length of its side, which we refer to as the *size* of the square.

In the offline setting, all square items are given in advance, and the algorithm can process them as a whole before placing any item into a bin. In particular, the algorithm can sort squares in the decreasing order of their sizes, and this comes in handy in many settings. In the online setting, the multi-set of items forms a *sequence* that is revealed in an online and sequential manner. Items are revealed one by one; when an item is revealed, an online algorithm has to place it into a square bin without any prior knowledge of forthcoming items. The decisions of algorithms are irrevocable.

Square packing has many applications in practice. One prominent application is cutting stuck where bins represent stocks (e.g., wood boards) and items are requests to squares of specific sizes. When requests arrive, an algorithm has to 'cut' the stock to provide the pieces that match the requests. This cutting process is equivalent to 'placing' items into bins. The goal of cutting stock is to minimize the number of stocks which also matches a square packing goal. We note that in many practical applications, requests arrive in an online manner and the stock should be cut without priory knowledge about the future requests. It is needless to say that the cutting process is irrevocable which gives an inherently online nature to these applications of square packing.

There has been a rich body of research around square packing. To our knowledge, all existing results except for our previous work on offline square packing [17], assume that squares are not allowed to rotate, that is, sides of square items should be parallel to the square bins. While this assumption makes combinatorial analysis of the problem easier, might cause a higher cost. As an example, consider an instance of the problem formed by n items of size 0.36. If we do not allow rotation, any bin can include at most 4 items, which results a total cost of $n/4$ for any algorithm. Allowing rotation, however, 5 items fit in each bin and we can reduce the cost to $n/5$ (see Figure 1). As a result, the number of required bins is decreased by $n/20$ which is a notable saving in practice (e.g., for cutting stock applications).

In this paper, we consider the online square packing problem with rotation which is defined as follows.

Definition 1 *In the online square packing with rotation, the input is a multi-set of squares (items) with sizes $S = \{x_1, ..., x_n\}$ where $0 < x_i \leq 1$. The goal is to pack these squares into the minimum number of squares of unit size (bins). In the offline setting, S is available since the beginning. In the online setting, S is revealed as a sequence $\sigma = \langle x_1, x_2, \ldots, x_n \rangle$ which is revealed in an online manner. At time-step t, the value of x_t is revealed and an online algorithm has to place a square of*

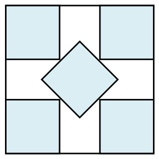


Figure 1: If all input items have length $\frac{\sqrt{2}}{2\sqrt{2}+1} \approx 0.35$, rotation allows packing 5 items per bin instead of 4.

[*]Department of Computer Science, University of Manitoba, Winnipeg, Canada, `shahin.kamali@umanitoba.ca`

[†]Department of Computer Science, University of Manitoba, Winnipeg, Canada, `nikbakhp@myumanitoba.ca`

size x $(1 \leq t \leq n)$. The decisions of the algorithm are irrevocable and are made without knowing the values of $x_{t'}$ for $t' > t$.

The asymptotic competitive/approximation ratio is the standard method for analyzing packing problems. We say an algorithm $\textsc{Alg}$ has a competitive ratio of c if there exists a constant c_0 such that, for all n and for all input sequences σ of length n, we have $\textsc{Alg}(\sigma) \leq c \cdot \textsc{Opt}(\sigma) + c_0$ where $A(\sigma)$ and $Opt(\sigma)$ denote the costs of A and Opt for processing σ, respectively, and are both arbitrarily large.

1.1 Related work

The 1-dimensional bin packing has been studied extensively in both offline and online settings (see, e.g., [14, 13, 6, 7]). In the 1-dimensional setting, each item has simply a size $\in (0, 1]$, and each bin has a capacity of 1. In the offline setting, the problem is NP-hard, and the best existing result is an algorithm that opens at most $\textsc{Opt}(\sigma) + O(\log \textsc{Opt})$ bins for placing a sequence σ [16]. In the online setting, the best existing algorithm has a competitive ratio of 1.578 [2]. Meanwhile, it is known that no online algorithm has a competitive ratio better than 1.54278 [3].

There are many ways to extend bin packing to higher dimensions (see [5] for a survey). Orthogonal packing square items into square bins is perhaps the most straightforward extension. In the offline setting, the problem is known to be NP-hard [18]. Bansal et al. [4] provided an APTAS for this problem (indeed, for the more general d-dimensional cubes). In the online setting, the best existing upper and lower bounds have improved a few times [19, 10]. The best existing algorithm has a competitive ratio of 2.0885 [8] while the best existing lower bound is 1.6707 [15].

In a previous work [17], we studied offline square packing when the rotation of items is allowed. We showed that, while the problem remains NP-hard, it admits an APTAS under a resource-augmentation setting, where bins of the algorithm have size $1 + \alpha$, for an arbitrary small $\alpha > 0$, while bins of $\textsc{Opt}$ have size 1. If resource augmentation is not allowed, the problem is likely $\exists\mathbb{R}$-hard [17, 1].

1.2 Contribution

We consider the online setting and provide an online algorithm that achieves a competitive ratio of 2.306 for the square packing problem. Our algorithm is based on classifying squares based on their sizes and placing squares of similar sizes tightly, possibly using rotations, in the same bins. This approach is previously used to introduce different families of *Harmonic algorithms* for bin packing in both one dimension and higher dimensions. The presence of rotations, however, make our classification and analysis different from the previous work.

2 Square-Rotate algorithm

In this section, we introduce our square packing algorithm called $\textsc{Square-Rotate}$.

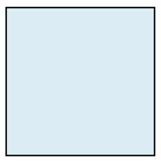
2.1 Item classification

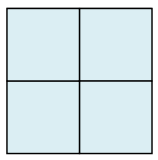
Similar to the Harmonic family of algorithms, we classify squares by the size of their side lengths (which we simply refer to as the *size* of the items).

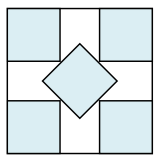
$\textsc{Square-Rotate}$ packs squares of each class separately from other classes. In total, there are 13 classes of squares (having more classes is possible but leads to none to small improvement of the final result). Square items with sizes in the range $(0, 0.1752]$ are in class 13. We refer to class 13 as the *tiny class*, and items that belong to this class are referred to as tiny items. We refer to items that belong to class $i \in [1, 12]$ as *regular items*. For each class $i \in [1, 12]$, the range of items in the class is specified as $(x_i, x_{i-1}]$ (for convenience, we define $x_0 = 1$). The values of x_i's are defined in a way that a certain number of items, denoted by S_i, of class i can fit in the same bin. The specific range of item sizes for each class $i \in [1, 12]$ and values of S_i is derived from the best-known or optimal results [12] on the congruent square packing problem [11], which asks for the minimum size $c(j)$ of a square that can contain j unit-sized squares. A scaling argument, where the container size is fixed to be 1, gives values of $u(j)$ when the goal is to pack j identical squares of maximum size $u(j)$ into a unit square.

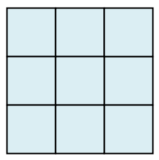
In Figure 2, it is specified how S_i items of the largest size in class i can fit into a square bin. Therefore, the scaled best-known values of $u(j)$ for $1 \leq j \leq 36$ can be derived from the figure. These scaled numbers give the specific ranges that we used for classifying items as follows: Items of class 1 have sizes in the range $(1/2, 1]$, and we have $x_1 = 1/2$. Note that exactly $S_1 = 1$ item of class 1 can fit in the same bin. For $i \in [2, 12]$, S_i is the number of items of size x_{i-1} that fit in the same bin. For example, for $i = 2$, we have $S_2 = 4$ because $x_1 = 1/2$, and 4 items of size 1/2 fit in the same bin. Moreover, x_i is defined as the largest value so that $S_i + 1$ items of size x_i cannot fit in the same bin. For example, we have $x_2 = 0.3694$ because according to Figure 2, $S_2 + 1 = 5$ squares of size 0.3694 cannot fit in the same bin.

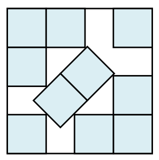
The respective range of items for each class, as well as the values of S_i, are presented in Table 1. For example, a square belongs to class 1, 2, or 12 if its side size is in the interval $(0.5, 1]$, $(0.3694, .5]$, or $(0.1752, 0.1779]$, respectively.

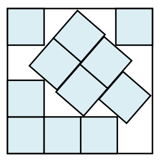

(a)
Class 1: $x \in (0.5000, 1.0000]$

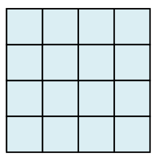

(b)
Class 2: $x \in (0.3694, 0.5000]$

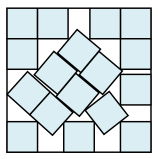

(c)
Class 3: $x \in (0.3333, 0.3694]$

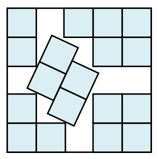

(d)
Class 4: $x \in (0.2697, 0.3333]$

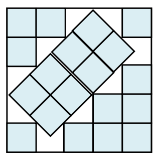

(e)
Class 5: $x \in (0.2579, 0.2697]$

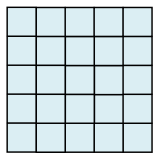

(f)
Class 6: $x \in (0.2500, 0.2579]$


(g)
Class 7: $x \in (0.2139, 0.2500]$


(h)
Class 8: $x \in (0.2073, 0.2139]$


(i)
Class 9: $x \in (0.2047, 0.2073]$


(j)
Class 10: $x \in (0.2000, 0.2047]$


(k)
Class 11: $x \in (0.1779, 0.2000]$

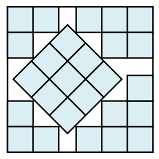

(l)
Class 12: $x \in (0.1752, 0.1779]$

Figure 2: Placement of regular square items of class $i \in [1, 12]$ in their respective bin. It is possible to pack i square items of class i into a single square bin [12].

Class	Side length x	S_i	Occupied Area	Weight	Density
1	$(0.5000, 1.0000]$	1	$> 1(0.250)$=0.250	1	< 4.000
2	$(0.3694, 0.5000]$	4	$> 4(0.136)$=0.544	1/4	< 1.838
3	$(0.3333, 0.3694]$	5	$> 5(0.111)$=0.555	1/5	< 1.801
4	$(0.2697, 0.3333]$	9	$> 9(0.072)$=0.648	1/9	< 1.543
5	$(0.2579, 0.2697]$	10	$> 10(0.066)$=0.660	1/10	< 1.515
6	$(0.2500, 0.2579]$	11	$> 11(0.062)$=0.682	1/11	< 1.466
7	$(0.2139, 0.2500]$	16	$> 16(0.045)$=0.720	1/16	< 1.388
8	$(0.2073, 0.2139]$	17	$> 17(0.042)$=0.714	1/17	< 1.400
9	$(0.2047, 0.2073]$	18	$> 18(0.041)$=0.738	1/18	< 1.355
10	$(0.2000, 0.2047]$	19	$> 19(0.040)$=0.760	1/19	< 1.315
11	$(0.1779, 0.2000]$	25	$> 25(0.031)$=0.775	1/20	< 1.290
12	$(0.1752, 0.1779]$	26	$> 26(0.030)$=0.780	1/26	< 1.282
13	$(0, 0.1752]$		> 0.702	$1.425x^2$	≈ 1.425

Table 1: A summary of item classification and details on item weights and densities, as used in the definition and analysis of Square-Rotate.

2.2 Packing regular items

For each class i ($1 \leq i \leq 12$), the algorithm has at most one active bin of type i. When a bin of type i is opened, it is declared as the active bin of the class, and S_i square *spots*, each having a size equal to the largest square of class i, are reserved in the bin. Upon the arrival of an item of class i, it is placed in one of the S_i spots of the active bin. If all these spots are occupied by previous items, a new bin of type i is opened. This ensures that all bins of type i, except potentially the current active bin, include S_i items.

2.3 Packing tiny items

For the last class, i.e., tiny items, the algorithm uses a different approach, proposed by Epstein and van Stee [9]. Briefly, it maintains at most one active bin for placing tiny items. When a bin is opened for these items, the algorithm reserves four square spots of size 1/2, i.e., the four squares of class 2 in Figure 2b. These square spots are used as bins for placing tiny items. Then, the algorithm chooses one of the innermost sub-bin squares that has enough space for the arrived item and repeats the procedure for the selected sub-bin until it cannot split any of the innermost sub-squares into four new ones with enough space for the item. At this step, the item is placed in one of those smallest sub-bins. When the next item arrives, if there is a sub-bin of the smallest possible size in which the item can fit, the algorithm places the item in that spot. Otherwise, the algorithm finds the smallest sub-bin that can fit the item and repeats the previous procedure to split it into the smaller sub-bins to reach an appropriate spot for the item. If no sub-bin with enough empty space is available in the bin, the algorithm closes the current bin and opens a new empty active bin for the item and applies the whole process from the beginning (see [9], for details). Note that the algorithm does not rotate any of the tiny items to pack them. Epstein and van Stee proved the following result, which we will use in our analysis later.

Lemma 1 *[9] Consider the square packing problem (without rotation) in which all items are of size at most $1/M$ for some integer $M \geq 2$. There is an online algorithm (as described above) that creates a packing in which all bins, except possibly one, have an occupied area of size at least $(M^2 - 1)/(M + 1)^2$.*

2.4 Algorithm's analysis

In this section, we prove a competitive ratio of at most 2.306 for our algorithm. We use a *weighting function argument*. For each item of size x, we define a weight $w(x)$ for the item and prove that: (1) the total weight of square items in each bin of the algorithm, except potentially a constant number of them, is at least 1, and (2) the total weight of items in each bin of an optimal packing is at most 2.306. If $w(\sigma)$ denote the total weight of items in an input sequence σ, then (1) implies that the number of bins opened by the algorithm is at most $w(\sigma) + c$, for some constant value of c, and (2) implies that the number of bins in an optimal packing is at least $w(\sigma)/2.306$. Therefore, the (asymptotic) competitive ratio of the algorithm would be at most 2.306.

Recall that all bins opened for squares of class i ($1 \leq i \leq 12$), except possibly the last active bin, include S_i squares. We define the weight of items of class i to be $1/S_i$. This way, the total weight of items in bins opened for all squares of classes 1 to 12, except possibly 12 of them (the last bin from each class), is exactly 1. Therefore, (1) holds for bins opened for regular items.

We define the weight of a tiny square of size x as $x^2/0.701 (= 1.425x^2)$. All tiny items are of size at most 0.1752. Therefore, by Lemma 1, the occupied area of all bins opened for tiny items (except possibly one of them) will be at least 0.701. This implies their total weight is at least $0.701/0.701 = 1$.

Table 1 gives a summary of the weights of items in different classes. From the above argument, we conclude the following lemma.

Lemma 2 *The total weight of squares in each bin opened by* Square-Rotate*, except possibly a constant number of them, is at least 1.*

The following lemma provides an upper bound for the total weight of items in a bin of the optimal offline algorithm (Opt). The proof works by case analysis and can be found in the appendix.

Lemma 3 *The total weight of items in every bin of* Opt *is less than* 2.306*.*

Provided with the above two lemmas, we can derive the main result of this section.

Theorem 4 *There is an online algorithm for the square packing problem with rotation problem which achieves a competitive ratio of at most* 2.306*.*

Proof. For an input σ, let $SR(\sigma)$ and $OPT(\sigma)$ denote the cost of Square-Rotate and Opt, respectively. Let $w(\sigma)$ denote the total weight of items of σ. Lemmas 2 implies that $SR(\sigma) \leq w(\sigma) + c$, where c is a constant independent of the length of σ. Meanwhile, Lemma 3 implies that $Opt(\sigma) \geq w(\sigma)/2.306$. From these inequalities, we conclude $SR(\sigma) \leq 2.306\ OPT(\sigma) + c$, which proves an upper bound 2.306 for the competitive ratio of Square-Rotate. □

References

[1] M. Abrahamsen, T. Miltzow, and N. Seiferth. Framework for ∃r-completeness of two-dimensional packing problems. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1014–1021. IEEE, 2020.

[2] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. A new and improved algorithm for online bin packing. In *26th Annual European Symposium on Algorithms (ESA)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[3] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. A new lower bound for classic online bin packing. *CoRR*, abs/1807.05554, 2018.

[4] N. Bansal, J. R. 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.

[5] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

[6] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In *Approximation algorithms for NP-hard Problems*. PWS Publishing Co., 1997.

[7] E. G. Coffman Jr., J. Csirik, G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: survey and classification. In P. M. Pardalos, D.-Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 455–531. Springer, 2013.

[8] L. Epstein and L. Mualem. Online bin packing of squares and cubes. *arXiv preprint arXiv:2105.08763*, 2021.

[9] L. Epstein and R. van Stee. Optimal online bounded space multidimensional packing. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 214–223, 2004.

[10] L. Epstein and R. van Stee. Online square and cube packing. *Acta Informatica*, 41(9):595–606, 2005.

[11] P. Erdős and R. L. Graham. On packing squares with equal squares. *Journal of Combinatorial Theory, Series A*, 19:119–123, 1975.

[12] E. Friedman. Packing unit squares in squares: A survey and new results. *The Electronic Journal of Combinatorics*, pages 1–24, 2000.

[13] G. Galambos and G. J. Woeginger. Online bin packing - a restricted survey. *ZOR*, 42:25–45, 1995.

[14] M. R. Garey and D. S. Johnson. Approximation algorithms for bin packing problems - a survey. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms in Combinatorial Optimization*, pages 147–172. Springer, New York, 1981.

[15] S. Heydrich and R. van Stee. Beating the harmonic lower bound for online bin packing. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[16] R. Hoberg and T. Rothvoss. A logarithmic additive integrality gap for bin packing. In *proc. the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2616–2625. SIAM, 2017.

[17] S. Kamali and P. Nikbakht. Cutting stock with rotation: Packing square items into square bins. In *International Conference on Combinatorial Optimization and Applications*, pages 530–544. Springer, 2020.

[18] J. Y. T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.

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

Appendix

Lemma 3 *The total weight of items in every bin of* Opt *is less than* 2.306*.*

Proof. We first define the *density* of an item of size x as the ratio between its weight and area, i.e., $w(x)/x^2$. Given the lower bound for the size of each square belonging to class i ($1 \leq i \leq 12$), we can calculate a lower bound for the density of each item in the class. For tiny items, the density is simply $1.425x^2/x^2 = 1.425$. Densities for all classes have been reported in Table 1.

Defining densities comes handy in the following case analysis to prove that the total weight of items in any bin B of an optimal packing is at most 2.306.

Case 1: First, assume there is no item of class 1 in B. Since the density of items of other classes are less than 1.838, even if B is fully packed with items of the largest density, the total weight of items cannot be more than 1.838 which is less than 2.306.

Case 2: In the second case, we assume there is one item x of class 1 (note that no two items of class 1 fit in the bin). Without loss of generality, we assume the size of x is $1/2 + \epsilon$, where ϵ is a small real value greater than zero. Clearly, a larger size for x does not increase the total weight of other items in B because it would leave less space to occupy more items in the bin (while the weight of x stays 1). Next, we consider all possible cases in which we have some items of class 2 and 3 together with x in B. As presented in Table 2, there will be 14 sub-cases to analyze. To see how we reach these 14 sub-cases, first note that it is not possible to accommodate 4 or more items of class 2 in addition to x in B (i.e., a total number of 5 or more items from these classes 1 and 2). This is because no five items with size larger than 0.3694 can fit in B [12]. A similar argument shows that we cannot have 6 or more items from classes 1, 2, and 3 together in a bin, otherwise we could accommodate 6 identical squares of size strictly larger than 0.3333 which is a contradiction to the fact that no six items of size larger than 0.3333 can fit in the same bin [12]. We can conclude that the 14 sub-cases summarized in Table 2 cover all possibilities for items of the first three classes in Case 2.

C1	**C2**	**C3**	**total weight of items of class $c \leq 3$ (W)**	**area occupied by items of class $c \leq 3$ (A)**	**remaining remaining Area ($A_r = 1 - A$)**	**total weight in remaining area ($W_r = A_r \times 1.543$)**	**total weight of items in the bin ($W + W_r$)**
1	0	0	1.00	> 0.250	< 0.750	< 1.157	< 2.157
1	0	1	1.20	> 0.361	< 0.639	< 0.986	< 2.186
1	0	2	1.40	> 0.472	< 0.528	< 0.815	< 2.215
1	0	3	1.60	> 0.583	< 0.417	< 0.644	< 2.244
1	0	4	1.80	> 0.694	< 0.306	< 0.472	< 2.272
1	1	0	1.25	> 0.386	< 0.614	< 0.948	< 2.198
1	1	1	1.45	> 0.497	< 0.503	< 0.776	< 2.226
1	1	2	1.65	> 0.608	< 0.392	< 0.605	< 2.255
1	1	3	1.85	> 0.719	< 0.281	< 0.434	< 2.284
1	2	0	1.50	> 0.522	< 0.478	< 0.738	< 2.238
1	2	1	1.70	> 0.633	< 0.367	< 0.566	< 2.266
1	2	2	1.90	> 0.744	< 0.256	< 0.395	< 2.295
1	3	0	1.75	> 0.658	< 0.342	< 0.528	< 2.278
1	3	1	1.95	> 0.769	< 0.231	< 0.356	< 2.306

Table 2: Fourteen possible cases in which we have a combination of items of class 2 (C2) and 3 (C3) together with one item x of class 1 (C1) in a single bin B. Here, "sum of weights (W)" and "sum of areas (A)" indicate, respectively, the total weight and area of items of the first three classes in B. "Remaining area" is the area left in the bin that is used for packing items of class 4 or higher. "Weight of items in the remaining area" is an upper bound for the total weight of items of class 4 or higher in B (these items have density no more than 1.543). Finally, "the total weight of items in the bin" indicate the sum of weights of all items (from all classes) in B.

According to Table 1, the density of items belonging to class i ($4 \leq i \leq 12$) as well as tiny items is at most 1.543 (which is the density of class-4 items). Using a similar argument made for Case 1, we suppose that, after placing a certain number of items of class 2 and 3 beside x in B, in each sub-case, we are able to completely fill the remaining empty space of B with the items of the maximum density 1.543. This makes us able to calculate an upper bound for the maximum total weight of items in B for each of the sub-cases. The resulting bounds for each sub-case can be found in the last column of Table 2, where the maximum upper bound among all sub-cases is 2.306, which happens when we have one item of class 1 in B together with 3 items of class 2 and one item of class 3.

As a result, in both Case 1 and Case 2, the total weight of items in B cannot be more than 2.306. $\square$