Acta Univ. Sapientiae, Informatica **7,** 1 (2015) 31–57

DOI: 10.1515/ausi-2015-0011

Bin packing with directed stackability conflicts

Attila BÓDIS
University of Szeged
email: bodis.attila@gmail.com

Abstract. The Bin Packing problem is a well-known and highly investigated problem in the computer science: we have n items given with their sizes, and we want to assign them to unit capacity bins such, that we use the minimum number of bins.

In this paper, some generalizations of this problem are considered, where there are some additional stackability constraints defining that certain items can or cannot be packed on each other. The corresponding model in the literature is the Bin Packing Problem with Conflicts (BPPC), where this additional constraint is defined by an undirected conflict graph having edges between items that cannot be assigned to the same bin. However, we show some practical cases, where this conflict is directed, meaning that the items can be assigned to the same bin, but only in a certain order. Two new models are introduced for this problem: Bin Packing Problem with Hanoi Conflicts (BPPHC) and Bin Packing Problem with Directed Conflicts (BPPDC). In this work, the connection of the three conflict models is examined in detail.

We have investigated the complexity of the new models, mainly the BPPHC model, in the special case where each item have the same size. We also considered two cases depending on whether re-ordering the items is allowed or not.

We show that for the online version of the BPPHC model with unit size items, every Any-Fit algorithm gives not better than $\frac{3}{2}$-competitive,

Computing Classification System 1998: F.2.2
Mathematics Subject Classification 2010: 68R05
Key words and phrases: Bin Packing Problem, conflicts, directed conflicts, Hanoi conflicts, unit size items, dynamic programming

when it is forbidden for the optimum to re-order the items, even if only 2 stackability classes, called Hanoi classes, are applied. This lower bound is generalized for arbitrary number of Hanoi classes. However, we also prove, that asymptotically the First-Fit algorithm is 1-competitive for this case.

Finally, we introduce an algorithm for the offline version of the BP-PHC model with unit size items, which has polynomial time complexity, if the number of the Hanoi classes and the capacity of the bins are constant.

1 Introduction

The Bin Packing Problem is one of the most known and investigated fields of the computer science, probably because it has several practical applications such as filling up boxes with certain products, loading trucks, etc. The problem is that we have n items given with their sizes, and we want to assign them to unit capacity bins so, that we use the minimum number of bins.

More formally, we have a set $N = \{1, 2, \dots, n\}$ of items, each item i has a size s_i, the bins have a capacity c, and we want to assign each item to one bin such that the total size of items in each bin is not exceeding c, and we use the minimum number of bins.

There are several variants of this problem in the literature including multi-dimensional cases, fragile objects, class-constrained items, coloured items, etc. We will summarize these models in Section 2.

In this work, we investigate a variant of the Bin Packing Problem, where additional constraints are occurred because of practical directed stackability conflicts, like some items are fragile and others are too heavy to pack them on each other. This kind of conflicts are quite common in industrial applications, especially in logistical ones.

After a short summary of the relevant existing variants of the Bin Packing Problem in Section 2, two new models are introduced for the mentioned directed stackability conflicts in Section 3. Then, the new models are compared with the corresponding model from the literature in Section 4.

We examined the complexity of a special case of the new models, where each item has unit size. This case is described, and the complexities are investigated for the two new models in Section 5.

2 Previous results

Unfortunately, in the real world applications the bin packing problem, just like a lot of computational problems, rarely happens in a pure way. Usually, there are additional constraints permitting or forbidding the assignment of specific items to specific bins based on the content of the bins, or determining the order of items in the bins. Various models are published for the Bin Packing Problem handling different kind of additional constraints mostly inspired by some kind of practical application. In this section, we will overview some of the existing models, especially the ones that are created for a similar approach to the current work.

First of all, we recall some definitions in connection with the approximation algorithms. For an algorithm $\mathcal{A}$, the solution of $\mathcal{A}$ for a given input X is denoted by $\mathcal{A}(\mathsf{X})$, and the optimal offline solution is usually denoted by $\mathsf{OPT}(\mathsf{X})$. The approximation ratio (called also competitive ratio in online cases) for a minimization problem is the minimal C value such that $\mathcal{A}(\mathsf{X}) \leq \mathsf{C} \cdot \mathsf{OPT}(\mathsf{X})$ is true for any X input. The asymptotic approximation (or competitive) ratio is defined similarly, but with $\mathsf{OPT}(\mathsf{X})$ approaching infinity.

Our first discussed generalization is the Bin Packing Problem with Fragile Objects, where each item have a fragility value in addition to its size, and the corresponding constraint is that the sum of the item sizes in a bin cannot exceed the fragility of any item in that bin. This problem is studied by for example Bansal et al. [2] and Clautiaux et al. [4]. The idea of handling the fragile objects is similar to the practical applications inspiring the current work, but we believe this model is quite difficult to extend to more than only one special object type, meaning the fragile ones.

In the Class Constrained Bin Packing problem, proposed by Shachnai and Tamir [22, 23], we have an additional parameter for every items defining the class of that item, and each bin has a storage capacity beside the load capacity, meaning that the number of different classes in the bins cannot exceed this storage capacity. In their paper, Shachnai and Tamir introduced a PTAS for the offline version of this problem. The online version was first investigated in a paper of the same authors [21]. Xavier and Miyazawa published results of application of this problem to Video-on-Demand services [26]. Further research on approximation algorithms for special cases of the problem was presented by Epstein et al. [11].

The class-constraint is limiting the number of different items in each bins, but this does not say anything about the order of the items in the bins. The next studied variant is exactly a constraint specifying this order. This is the

Colored Bin Packing problem studied recently in some papers [1, 3, 7]. In this model, the items have a color value, and the additional constraint is that one cannot put two items with the same color successively in the same bin. Böhm et al. [3] showed that the classical algorithms First-Fit, Best-Fit and Worst-Fit are not constant competitive. A special case is investigated by Balogh et al. [1], when there are only two colors, black and white, and they have shown a lower bound about 1.7213 on the asymptotic competitive ratio for any online algorithm.

There is a model which is more similar to the one discussed in this work, than the above ones, this is the Bin Packing Problem with LIB ('Largest In Bottom') constraints, which has been also investigated by several authors [10, 9, 16, 17]. This additional constraint means that one cannot put an item on another one with smaller size. Epstein [10] proved that First-Fit gives not better than 2.5 competitive for the online case, which was improved by Dósa et al. [9] to about 2.1666, and they also mentioned a model of Generalized LIB constraint, where the constraint is defined by an undirected incompatibility graph based on the sizes of the items, and adjacent items cannot be packed into the same bin.

The Bin Packing Problem with Conflicts (BPPC) model, investigated by several authors, like Jansen and Öhring [14], Jansen [13], Sadykov and Vanderbeck [20] etc., is very similar to the Generalized LIB constraint in the sense that both models use a conflict graph $G = (V, E)$, where E is the set of edges so, that if $(i, j) \in E$, then the items i and j cannot be packed into the same bin.

The Bin Packing Problem with Conflicts model is very general as basically any kind of graph can be used as a valid conflict graph. There are papers also about special conflict graphs, for example McCloskey and Shankar published results for the case of clique-graphs [19], Jansen and Öhring [14], and also Epstein and Levin [12] studied perfect conflict graphs, and bipartite graphs. Khanafer et al. [15] investigated the two-dimensional variant of this problem.

Finally, our contribution to the field of constrained bin packing is that, in this work, we will introduce a new stackability constraint considering the order of the items, and we will generalize the BPPC model with directed edges.

3 Model definition

In this section, we show some of the real world applications where the undirected conflict graph of the BPPC model is not appropriate, and we define a

new model, which is partly based on the practical solutions, handling these special stackability constraints. Then, we introduce another model, which is generalizing all of the mentioned conflict models.

3.1 Problem definition

As we already mentioned, the computational problems rarely happen purely in the real-world applications, and this is true for the Bin Packing Problem as well. At several fields of the everyday life, we want to assign our items into bins such that some items cannot be packed on each other. This problem is partly handled by the existing BPPC model, but this is not able to handle the case where the conflict is directed. This direction means that the conflict occurs only when the items are packed into the same bin in a given order. For example, there are some fragile products, and we do not want to put them at the bottom of the bins, but we can put them on the top of the bins independently from the content of the bins.

These stackability problems are present in several logistical approaches, like palletisation, when heterogeneous unit loads are expected to be stable enough for transportation, or truck loading, where we want to consider the unloading order. This also occurs in the everyday life, when we want to pack into minimum number of bags at the shop, such that we do not want to put the milk carton on the tomatoes but it can be packed on the potatoes, or even the tomatoes can be placed on the cartons.

Many other examples could be written for the practical applications, where the conflicts are directed. However, this kind of constraints is not handled in the BPPC. Actually, when we have a fixed order of the items, then we can define the edges such that this constraint is taken into consideration, as we will show this in Section 4., but this is not the case for the general problem.

3.2 Bin packing problem with hanoi conflicts (BPPHC)

What is common in the mentioned applications is that there are constraints between the items specifying the order of the items in the bins. These constraints are describing some kind of stackability between the items, which can be defined by precedences or stability expectations, but in either cases it means that some items can or cannot be put on other ones. This definition is very similar to the one appearing at the mathematical game called Tower of Hanoi, in the sense that there are also specific items that cannot be put on other ones. So we will name these restrictions to Hanoi conflicts in this work.

Based on the observation of the real-world approaches, we introduce a generalized bin packing problem handling these stackability constraints. The generalization appears in the description of the items. In our model, each item i is described not only by its size s_i, but also by an additional value determining its Hanoi property.

Definition 1 *The Hanoi property of an item* i *is* $h_i \in H = \{1, \ldots, m\}$*, where* H *is the set of possible Hanoi properties.*

Definition 2 *The Hanoi conflict is that one cannot put any item on another one with higher Hanoi property. More formally, if item* i *is assigned to a bin earlier than item* j*, then one can put item* j *into the same bin as item* i *if and only if* $h_i \leq h_j$*.*

Using the definition of the Hanoi conflict, the Bin Packing Problem with Hanoi Conflicts is defined as following. We have a set $N = \{1, 2, \ldots, n\}$ of items, each item i has a size s_i and a Hanoi property h_i, the bins have a capacity c, and we want to assign each item to one bin such that the total size of items in each bin is not exceeding c, the Hanoi conflicts are considered, and we use the minimum number of bins.

In order to describe a possible mathematical formulation for this model, we have to introduce some indicator variables, based on the formulation for the standard BPP by Martello and Toth [18]. Let denote $x_{i,j}$ if item j is assigned to the bin i, or not, and let y_i denote if the bin j is used or not. More precisely:

$$x_{i,j} = \begin{cases} 1 & \text{if item } j \text{ is assigned to bin } i, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

$$y_i = \begin{cases} 1 & \text{if bin } i \text{ is used,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Then the model for the BPPHC can be defined as the following.

$$\text{minimize } z = \sum_{i=1}^{n} y_i \tag{3}$$

$$\text{subject to } \sum_{j=1}^{n} s_i x_{i,j} \leq c y_i \qquad i \in N = \{1, \ldots, n\} \tag{4}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \qquad j \in N \tag{5}$$

$$\left(\sum_{k=1}^{n} x_{k,i} \cdot x_{k,j} \right) (i - j)\,(h_i - h_j) \geq 0 \qquad i \in N,\ j \in N \tag{6}$$

$$y_i \in \{0, 1\} \qquad i \in N \tag{7}$$

$$x_{i,j} \in \{0, 1\} \qquad i \in N, j \in N. \tag{8}$$

This formulation differs from the one for the standard Bin Packing Problem only in the extra constraint (6). The sum is defining that item i and j are assigned to the same bin taking the value of either 1 or 0. The second and the third factors are assuring the consideration of Hanoi conflict between the two items: the sign of them must be the same, meaning that the item assigned later to the bin must have the higher Hanoi property.

We note that this formulation is not linear, because we take the product of variables in the constraint (6). We also mention that this constraint is assuming that item i is the i^{th} item being assigned to a bin, which means that re-ordering the items is not allowed. We will discuss this additional assumption in detail in Subsection 5.2.

3.3 Bin packing problem with directed conflicts (BPPDC)

Obviously, our Bin Packing Problem with Hanoi Conflicts model is somehow connected to the BPPC model, as the Hanoi constraints can be represented as a conflict graph. However, while in the BPPC an edge (i, j) means item i and item j cannot be assigned to the same bin, in the BPPHC model this edge means only that item j cannot be assigned to the same bin as item i later than item i, but it can be assigned to that bin earlier. Thus, in our model the graph is directed according to the Hanoi conflicts.

Since the Hanoi conflicts imply a directed conflict graph, let introduce the Bin Packing Problem with Directed Conflicts (BPPDC) model. We have the set of items $N = \{1, 2, \ldots, n\}$, each item i has a size s_i, the bins have a capacity

c, and we have a directed conflict graph $G = (V, E)$, where E is the set of edges so, that if $(i, j) \in E$, then the item i cannot be packed on top of item j in the same bin. We want to assign each item to one bin considering this conflict graph such that the total size of items in each bin is not exceeding c, and we use the minimum number of bins.

The mathematical formulation of this model can be created using the idea of the BPPHC model.

$$\text{minimize } z = \sum_{i=1}^{n} y_i \tag{9}$$

$$\text{subject to } \sum_{j=1}^{n} s_j x_{i,j} \leq c y_i \qquad i \in N = \{1, \ldots, n\} \tag{10}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \qquad j \in N \tag{11}$$

$$\left(\sum_{k=1}^{n} x_{k,i} \cdot x_{k,j} \right) (i - j) \geq 0 \qquad (i, j) \in E \tag{12}$$

$$y_i \in \{0, 1\} \qquad i \in N \tag{13}$$

$$x_{i,j} \in \{0, 1\} \qquad i \in N, j \in N \tag{14}$$

The notes for the model of the BPPHC are relevant for this model as well: this is not a linear formulation, and re-ordering the items is not allowed.

4 Connection to the bin packing problem with conflicts

In this section, firstly, we will show the connections between the mentioned three conflict models. Then, we will show the importance of the order of the input items, and we will analyse the models considering this order.

The connection of the models is briefly visualized, and also the theorems describing that certain connection is shown, on Figure 1.

4.1 Connections between the three conflict models

In this subsection, we will show that the BPPDC model is the most general among the three models, and that the BPPDC and the BPPHC models are equivalent under certain conditions.

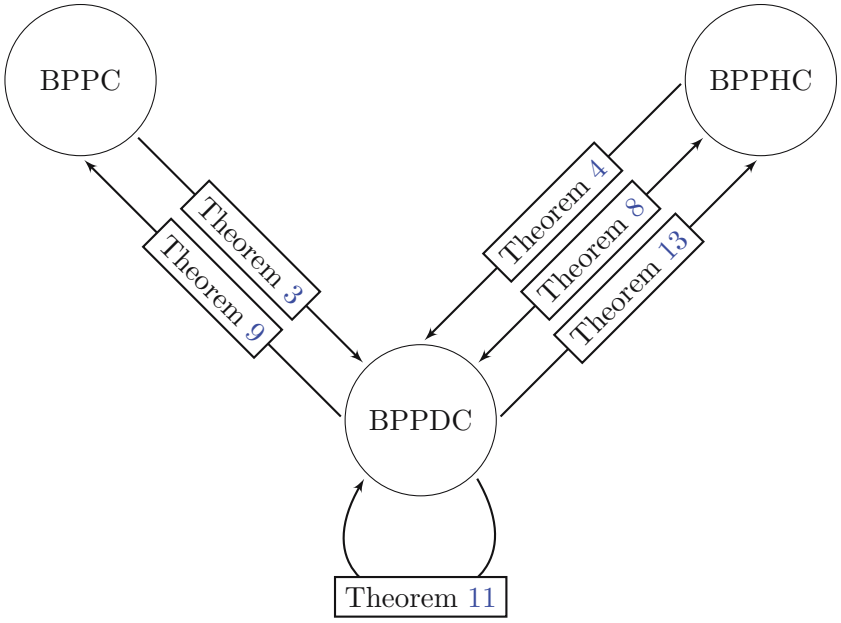

Figure 1: The connection of the three models and the corresponding theorems

First, we present a theorem about the connection of the BPPC and the BPPDC models.

Theorem 3 *The BPPDC model is a generalization of the BPPC model.*

Proof. We show that for any $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ undirected conflict graph, we can create a $\mathsf{G}' = (\mathsf{V}, \mathsf{E}')$ directed conflict graph so, that the corresponding BPPC and BPPDC models are equivalent.

This can be achieved quite simply by generating E' from E in the following way: $\mathsf{E}' = \{(\mathsf{i}, \mathsf{j}), (\mathsf{j}, \mathsf{i}) \mid (\mathsf{i}, \mathsf{j}) \in \mathsf{E}\}$, which means that we take two directed edges for each undirected edge. It can be easily seen that this will result exactly the same problem, because, if item i and j cannot be assigned to the same bin in the BPPC model, then one cannot pack them into the same bin in any order, so we have to take two directed conflicts. □

Now, let investigate the connection of the two new models: the BPPHC and the BPPDC models.

Theorem 4 *The BPPDC model is a generalization of the BPPHC model.*

Proof. We show that we can create a $G = (V, E)$ directed conflict graph so, that each of the Hanoi conflicts are covered by the edges meaning that, two items are in Hanoi conflict if, and only if, there is an edge between them with proper direction in the resulting G graph.

Let the Hanoi constraint be given by item i and j, and $h_i < h_j$, then we take an edge (i, j) into the directed conflict graph. It is trivial, that with this transformation we get an equivalent BPPDC model for any BPPHC input, because the Hanoi conflict of the item i and j represents that the item i cannot be assigned to the same bin as the item j later, than the item j, and the taken edge denotes exactly the same in the BPPDC model. □

Our next theorem will be about the equivalency of the BPPHC and the BPPDC models, but this is realized only under some certain conditions for the type of the directed conflict graph. So before saying that theorem, we have to recall some definitions in connection with the directed graphs.

Definition 5 *If in a directed graph* $G = (V, E)$ *having edges* $(u, v) \in E$ *and* $(v, w) \in E$ *implies also having the edge* $(u, w) \in E$*, then the graph is called transitive.*

We also have to define a special type of the graphs, for which the equivalency will occur.

Definition 6 *A directed graph* $G = (V, E)$ *is called a transitive path, if it is a path with additional edges such that* G *is transitive.*

Definition 7 *A directed graph* $G = (V, E)$ *is called a Hanoi graph, if the graph containing its independent sets as vertices is a transitive path. With other words,* G *is a Hanoi graph if it can be generated from a transitive path* $G' = (V', E')$ *in the following way. We create an independent set of vertices* $V_{i'} \subseteq V$ *for every vertex* $i' \in V'$*, and we take the edges such that* $E = \{(i, j) \,:\, i \in V_{i'},\ j \in V_{j'},\ (i', j') \in E'\}$.

Now, we can present our equivalency theorem.

Theorem 8 *A directed conflict graph is generated by Hanoi conflicts, if and only if, it is a Hanoi graph.*

Proof. First, we prove that the directed conflict graph generated by Hanoi conflicts must be a Hanoi graph.

It is easy to see by the definition of Hanoi conflicts (Definition 2), that if we consider the directed conflict graph $G = (V, E)$, created with the algorithm described in the proof of Theorem 4 from the Hanoi properties themselves, meaning we have exactly one item for every property, then G is a transitive path. Then, based on the definition of the Hanoi graph (Definition 7), we only have to create the independent sets of vertices from the items with equal Hanoi property, which is possible because there is not conflict between these items. So, we get a Hanoi graph $G' = (V', E')$. This means that we can always create a directed conflict graph from the Hanoi conflicts so, that it is a Hanoi graph.

Now, we have to prove that this directed conflict graph is unique. This is proved indirectly. Let assume that there is another directed conflict graph $G'' = (V'', E'')$, which defines exactly the same conflicts as the Hanoi conflicts, and which is different from G'. As the items are the same, $V' = V''$ must occur, so the difference can appear only in the edges, which is possible in two cases:

1. If there is an edge e such that $e \in E'$, but $e \notin E''$, then G'' skips a conflict, which is defined by the Hanoi conflicts, so it is not valid. This is a contradiction.

2. If there is an edge e such that $e \notin E'$, but $e \in E''$, then G'' has an extra conflict, which is not defined by the Hanoi conflicts, so it is also not valid. This is a contradiction, as well.

So we get, that the generated directed conflict graph is unique and it is Hanoi graph indeed.

Secondly, we prove that any $G = (V, E)$ Hanoi graph can be generated by Hanoi conflicts. For this, we show that one can set the Hanoi properties of the items such that, the resulting BPPHC model is equivalent to the original BPPDC model.

We can give an algorithm for this transformation. First, we create the transitive path $G' = (V', E')$ from the independent sets of vertices of G. Actually, we make this inversely as in the definition of the Hanoi graph (Definition 7). Then we have to set the Hanoi properties for every vertices in G' according to the edges. For this, we define for every vertex $i' \in V'$ the set of predecessors $P_{i'}?\{j' \mid (j', i') \in E'\}$. Then for every item i' with $|P_{i'}| = 0$, actually there is only 1 such item because G' is a transitive path, let $h_{i'} = 1$, and for every item i' with $|P_{i'}| > 0$ let $h_{i'} = 1 + \max_{j' \in P_{i'}} h_{j'}$. The resulting Hanoi properties are correct, because any item can be packed on the items without predecessors, and the other items cannot be packed below their predecessors, because of the definition of the Hanoi conflicts (Definition 2). With this algorithm, we set

the Hanoi properties only for the independent sets, but we need them for all items. As the items in the same independent sets are not in conflict, we can set the same Hanoi property for every item in the same independent set. So, we managed to show that G can be generated by Hanoi conflicts. □

So, we have proved, that the equivalency between the BPPHC and the BPPDC models is really existing under the quite strict condition of having a Hanoi graph.

4.2 Importance of the order of the items

By definition, the effect of the Hanoi conflicts are highly dependent on the order of the items. For example, given an item i and j, if $h_i < h_j$ and item i is assigned earlier to a bin, then item j can be assigned to the same bin. However, if item j is arrived earlier, then item i cannot be assigned to the same bin as item j. This means, that one can change if two items are assigned to the same bin by changing only the order of the items. This kind of importance of the input order is not apply in the BPPC model, because the conflict graph defines the conflicts independently from the order.

In this section, we will show that for a fixed order of the items, any directed conflict graph has an equal undirected conflict graph. Also, we will show that with fixed order of items there is a special type of the directed conflict graphs that can be transformed into Hanoi conflicts.

Theorem 9 *Let be given a fixed order of the input items. Then for any directed conflict graph one can find an undirected conflict graph defining exactly the same conflicts.*

Proof. Let (i, j) be an edge in the directed graph, which means that the item i cannot be assigned to the same bin as the item j later, than item j. Considering the given order of the items, there are two cases we have to investigate.

If item i is arrived earlier, then there is no way to assign this to a bin later, than the item j, which is still not arrived. This means, that we can ignore this edge, so there will not be (i, j) edge in the undirected graph.

If item j is arrived earlier, then item i definitely cannot be assigned to the same bin as item j, because of the (i, j) directed edge. Thus, we have to add an (i, j) edge to the undirected graph, and, according to the given order, this has exactly the same meaning as the directed edge. □

Now, we observe the general and the acyclic directed conflict graphs for this case. For this, we need the definition of the directed acyclic graph.

Definition 10 *A directed acyclic graph (DAG) is a directed graph without directed cycles, meaning that there is no way to get back to a vertex through a path following the directed edges* [24].

Theorem 11 *Let be given a fixed order of the input items. Then for any directed conflict graph* $G = (V, E)$ *one can find an acyclic directed conflict graph* $G' = (V, E')$ *defining exactly the same conflicts.*

Proof. Let C be a cycle in G. Considering the first vertex of C in the order, we get that the edge starting from this vertex must go to a vertex arrived later, so we can ignore this edge, because having a conflict with an item not arrived yet is irrelevant.

This means, that, according to the given input order, we can ignore at least one of the edges from any cycle. Thus, there exists an acyclic directed conflict graph for any directed conflict graph. □

Before we can introduce our theorem about under which conditions is it possible to transform a BPPDC model to a BPPHC model for a given order of the input items, we still have to recall the definition of the (weakly) connected directed graphs.

Definition 12 *A directed graph is called (weakly) connected, if removing the directions of the edges and considering them as undirected ones, results to a connected undirected graph.*

Theorem 13 *Let be given a fixed order of the input items and a directed conflict graph. Assuming that the items of each (weakly) connected components of the graph is arrived in continuous blocks, meaning there is no item from another component between any two items of the same component, and every such component is a Hanoi graph, one can set the Hanoi properties for the items such that, according to the given order, the resulting BPPHC model is equivalent to the original BPPDC model.*

Proof. In the second part of the proof of Theorem 8, we have shown that any directed conflict graph, which is a Hanoi graph, can be generated by Hanoi conflicts, and we described an algorithm to set the Hanoi properties based on that graph. To prove the current theorem, we have to expand that algorithm so, that it can handle multiple (weakly) connected components.

Let denote the (weakly) connected components of $G = (V, E)$ by $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_m = (V_m, E_m)$. Without loss of generality, we can assume that the components are arriving in the order of their indices. We

note, that we use here the fact that the items of each components are arrived in a continuous block. This is important, because, if the components are merged in the order, then we cannot disjoin them to set the Hanoi properties correctly.

Using this notation, we can define Algorithm 1, which makes the necessary transformation. The $\mathtt{set\_properties\_for\_Hanoi\_graph}(G_i,\ \mathtt{start})$ function is actually the algorithm in the second part of the proof of Theorem 8 with a little change, that the lowest Hanoi property set for any item of G_i is at least $\mathtt{start}$, and it also updates the value of $\mathtt{globalmax}$ by the maximal property appearing in G_i.

Algorithm 1: Algorithm setting the Hanoi properties for the items according to a directed conflict graph and a fixed input order

```
globalmax := 0 ;
foreach G_i in G do                          // in the fixed input order
    start := globalmax;
    set_properties_for_Hanoi_graph(G_i, start);
end
```

Now, we have to prove the correctness of this algorithm. Obviously, the Hanoi properties are set correctly inside each components as shown in the proof of Theorem 8. We have to prove only that the Hanoi properties are appropriate between the components, as well. In this case, appropriateness means that for any two items from two different components, the item arrived later has the greater Hanoi property. More formally, $\forall i \in G_t, \forall j \in \{G \backslash G_t\} : h_i \leq h_j$ if and only if item i is arrived earlier than item j. This is ensured by the usage of the $\mathtt{globalmax}$ variable so, that the items of the currently visited components get the greatest Hanoi properties, and we are visiting the components in the fixed order. So, the items of the earlier components get the lower Hanoi properties. □

4.3 Summary of the connections of the models

In this section, we reported our results on investigating the connections between the Bin Packing Problem with Conflicts (BPPC) from the literature, and our two proposed models: the Bin Packing Problem with Hanoi Conflicts (BPPHC) and the Bin Packing Problem with Directed Conflicts (BPPDC). We have shown that the last one is the most general as the other two models can be reduced to this one. Then we proved an equivalency theorem about BP-

PHC and BPPDC stating that a directed conflict graph is generated by Hanoi conflicts, if and only if, it is a Hanoi graph. This theorem actually characterizes the BPPHC model compared to the BPPDC model.

Then we discussed the importance of the order of the items, because, unlike the BPPC model, this is essential for the two new models. We examined possible transformations between the models for fixed order. We pointed out that for a fixed order, a directed conflict graph can be transformed into an undirected conflict graph. Furthermore, we presented an algorithm transforming a 'special type' of BPPDC input into BPPHC input that is this algorithm can set the Hanoi properties for the items based on the directed conflict graph. The 'special type' means here that the (weakly) connected components of the directed conflict graph must arrive in continuous blocks, meaning separately from the other components, in the fixed order, and each of these components must be a Hanoi graph.

5 Variants with unit size items

In this section, we will consider the above described models with a further assumption that each item have the same unit size. This case is also investigated for other variants of the Bin Packing Problem and other packing problems as well. Several authors, like Coffman et al. [5, 6], studied the ordinary Bin Packing Problem with discrete item sizes, which is a similar, but weaker assumption for the sizes. Shachnai and Tamir [21] proposed algorithms for the Class-Constrained Bin Packing Problem with unit sizes.

This case has practical applications, too, mainly in logistics. For example, we have to load truck with equal size pallets and we have to consider the order of unloading, meaning we cannot put some pallets in front of others. Also, this case can occur in several approaches, where different products have boxes with same size and the conflicts are based on some logical conditions such as one item has to be used earlier than others.

For this variant, we slightly modify our models such that $s_i = 1$ for every item i, and $c > 1$ that is the capacity of the bins is higher than 1. We will call the modified versions of the BPPHC and the BPPDC models BPPHCU and BPPDCU models respectively, adding the 'and Unit size items' suffix for the names of the models.

As shown in the Subsection 4.2, the order of the items is crucial, so we will consider two cases based on whether the algorithm can or cannot change the order of the items.

5.1 Ordering is allowed

If one can change the order of the input, then the problems usually becomes easier, and this is the case for our problems, as well. In this, part, we will introduce algorithms for both new models, when it is allowed to change the input order.

5.1.1 BPPHCU with ordering

For the case of Hanoi conflicts, if one can re-order the items, the problem becomes absolutely easy. Actually, in this case, there is no real effect of the Hanoi conflicts for the result, because we can always sort the items per bin such that we get a valid solution. This is concluded in the next theorem, where we define an algorithm based on Next Fit giving optimal solution.

Theorem 14 *The following algorithm gives optimal solution for the BPPHCU model.*

Firstly, we pack the items into bins using the Next-Fit algorithm not considering the Hanoi conflicts at all. Then we sort the items inside each bins separately such that the Hanoi conflicts do not occur, meaning the items with lower Hanoi property will be the earlier in the item-list of each bin.

Proof. Obviously, Next-Fit gives optimal solution for the ordinary Bin Packing Problem with Unit sizes. Then, we have to prove only, that we can sort the items in each bins such that the Hanoi conflicts are avoided. This is also quite trivial, because one can always sort positive integers, that is the Hanoi properties, into non-decreasing order. □

5.1.2 BPPDCU with ordering

The BPPDCU model usually also becomes relatively easy, if we can re-order the items, because we can make a topological order of the items resolving exactly the directed conflicts. However, the topological sort is possible only, if the graph is acyclic, meaning if there are items that cannot be packed into the same bin, independently from their order in the bin, then the problem is still not trivial.

In this work, we consider only the acyclic graphs for this problem, and the next theorem defines an efficient algorithm, similar to the one described in Theorem 14, to find the optimal solution in this case.

Theorem 15 *The following algorithm gives optimal solution for the BPPDCU model.*

Firstly, we pack the items into bins using the Next-Fit algorithm not considering the directed conflict graph at all. Then we sort the items inside each bins separately corresponding to the topological order, meaning the items appearing earlier in the topological order will be the earlier in the item-list of each bin.

Proof. The correctness of this algorithm trivially comes from the proof of Theorem 14 and the definition of the topological order. □

5.2 Ordering is forbidden (BPPHCU)

As we already described in Section 4.2, the order of the items is crucial in connection with the directed conflicts including the Hanoi conflicts as well. We have seen in the previous subsection that the problem becomes kind of easy if we have the ability to re-order the input items. However, this is not possible in several cases. In logistics, this is forbidden usually because of lack of puffer spaces. For example, we have to put the pallets on a truck when it is ready for transport, because we do not have enough space in the warehouse to wait all the pallets. This is similar to the definition of the online problems, but sometimes the full list of pallets is available, so the online and offline variants of the problem is also interesting.

In this subsection, we will first investigate the online case, meaning we get each item one-by-one, and we do not know anything about the later ones. We show a lower-bound on the approximation ratio of the Any-Fit algorithms. Then, we will prove that asymptotically the First-Fit algorithm, modified for Hanoi conflicts, is 1-approximation. Finally, we will show an offline optimal algorithm with polynomial time complexity if the number of different Hanoi properties and the capacity of the bins are considered to be constants.

5.2.1 Online case

In this part, we investigate the online algorithms for the BPPHCU model, when the offline optimum is restricted such that the offline algorithm can neither re-order the items, so it has to pack the items in their incoming order.

First-Fit algorithm is widely investigated for different variants of the Bin Packing Problem. For the standard version, it was proved by Ullman [25] that its asymptotic approximation ratio is 1.7, and Dósa and Sgall [8] proved, that the absolute approximation ration is the same. We have to modify this algorithm to handle the Hanoi conflicts: the current item is assigned to the

first not-filled bin, where the top-item, meaning the last item packed into that bin has a lower or equal Hanoi property.

First-Fit is a member of a group of algorithms, called Any-Fit algorithms defined by a general idea. Any-Fit algorithms always assign the current item to any of the already open, incomplete bins, if it is possible, otherwise they open a new bin for this item.

In the next theorem, we present a lower-bound for the Any-Fit algorithm, when $m = 2$, meaning there are only 2 Hanoi classes.

Theorem 16 *Every Any-Fit algorithm gives not better than $\frac{3}{2}$-approximation for the BPPHCU model with $m = 2$.*

Proof. To prove this theorem, we show an example, where the approximation ratio of the Any-Fit algorithm is $\frac{3}{2}$.

Let $I = [1, 1, 1, 2, 2, 2, 2, 1]$ be the list of input items, where the items are given only with their Hanoi properties because we have unit sizes, and the capacity of the bins is 4.

This is packed by Any-Fit as follows:

$$AF(I) = \{1, 1, 1, 2\}, \{2, 2, 2\}, \{1\}.$$

OPT can pack the items, like following:

$$OPT(I) = \{1, 1, 1, 1\}, \{2, 2, 2, 2\}.$$

We can see, that Any-Fit packed the items into 3 bins, while the optimum can solve that problem instance with only 2 bins. So, the approximation ratio in this case is $\frac{AF(I)}{OPT(I)} = \frac{3}{2}$ □

This lower bound can be generalized to arbitrary number of Hanoi classes using the same structured of input. This result is stated in the next theorem.

Theorem 17 *Let assume m is the number of Hanoi classes, c is the capacity of the bins, and $\lfloor x \rfloor$ denotes the integer part of x. If c is a divisor of m, then every Any-Fit algorithm gives not better than $\frac{\left\lfloor \frac{m}{c} \right\rfloor (c-1) + m}{m}$ -approximation, otherwise they are not better than $\frac{2m - \left\lfloor \frac{m}{c} \right\rfloor - 1}{m}$ -approximation for the BPPHCU model.*

Proof. To prove this theorem, we show an input structure for arbitrary m and c, where the approximation ratio of the Any-Fit algorithms is exactly the one mentioned in the theorem.

Let I be the list of input items as follows:

$$I = [1, 1, \ldots, 1;; 2, 2, \ldots, 2;; \ldots;; m, m, \ldots, m;; m, m-1, \ldots, 2, 1]\,.$$

Where the items are given only with their Hanoi properties because we have unit sizes, and the capacity of the bins is c. In the input above, the semicolons denote a separator of input blocks such that, the number of items in each block, except the last one, is $c-1$.

As the last block contains exactly one item for each Hanoi property, an optimal solution, denoted by $\mathsf{OPT}(I)$, is to fill 1 bin for each Hanoi property, because there are exactly c items for each of them, thus $\mathsf{OPT}(I) = m$.

Now, let consider how the Any-Fit algorithms work on such an input. As the items can be packed on each other in the order of input until the last block, Any-Fit packs these items on each other until the bin is filled, then open a new bin and do the same, which is basically a simple Next-Fit method. However, when the algorithm reaches the last block, then it has to pack each item into a new bin, so we get a lot of bins with only 1 item.

Firstly, let investigate the blocks containing $c-1$ items. As the capacity of the bins is c, every bin will contain items from the consecutive blocks so, that, when we iterate through the input, the packed bins will always contain one more item from the next block, than the previous bin. This is clearly seen on an example, where $c = 4$. In this case, Any-Fit packs these input-blocks as following:

$$\mathsf{AF}(I) = \{1,1,1,2\}, \{2,2,3,3\}, \{3,4,4,4\}, \{5,5,5,6\}, \{6,6,7,7\}, \ldots$$

This means that after c blocks $c-1$ bins, called bin-block, are filled and the next block starts with a new empty bin. Based on this property of Any-Fit, we can divide into 3 groups the packed bins according to which part of the input items are packed into them. The bins that are packed with items from the blocks containing $c-1$ items have 2 groups: the bins that are in a complete bin-block, meaning having exactly $c-1$ bins, and the ones that are in the remainder bin-block. The third group of bins contains the ones being packed with items from the last block. We note, that the first item of the last block can be packed on the items of the previous block, and, in this case, this bin is also in the third group.

So, to determine the bins used by the Any-Fit algorithms, we have to count them in each groups. The number of bins in the first groups is quite trivial considering the already mentioned fact, that after every c blocks $c-1$ bins are filled. As the number of such input blocks is m, the number of bins in the first groups is $\left\lfloor \frac{m}{c} \right\rfloor (c-1)$.

The third group contains exactly m bins, because every item in the last block is placed into a new bin. We note again, that the first item is possibly packed into the same bin as the last item of the previous block, and we count this bin also for the third group.

The second group, that is the remainder bin-block, is a bit more complicated. We have to consider two cases. If c is a divisor of m, then there are not any bin in the remainder block, so the number of bins in the second group is 0. Otherwise, let denote r the number of remained input blocks for this part. Then $r = m - 1 - \left\lfloor \frac{m}{c} \right\rfloor c$, which is similar to the definition of the remainder of the division $\frac{m}{c}$, except we decrease this by one, because the last block is counted in the third group. As $r < c$, the number of bins used to pack these items is exactly r, because we should have at least c blocks to save one bin.

So by summarizing the number of bins in the 3 groups, we get that if c is a divisor of m, then the approximation ratio is the following:

$$\frac{AF(I)}{OPT(I)} = \begin{cases} \dfrac{\left\lfloor \frac{m}{c} \right\rfloor (c-1) + m}{m} & \text{if } c \text{ is a divisor of } m \\ \dfrac{\left\lfloor \frac{m}{c} \right\rfloor (c-1) + m - 1 - \left\lfloor \frac{m}{c} \right\rfloor c + m}{m} & \\ = \dfrac{2m - \left\lfloor \frac{m}{c} \right\rfloor - 1}{m} & \text{otherwise.} \end{cases}$$

□

Although, we have seen that Any-Fit has at least $\frac{3}{2}$-approximation ratio even for $m = 2$, asymptotically the situation is much better for the First-Fit algorithm. Before we show our results about this, firstly we have to introduce an intermediate result about the incomplete bins (the ones that are not filled up totally) during the execution of this algorithm.

Theorem 18 *There are no two incomplete bins having top-items with the same Hanoi property at any time during the execution of the First-Fit algorithm.*

Proof. This is proved by induction on the number of items arrived (let denote this by i, and the Hanoi property of the last item by h_i):

1. If $i = 1$, then we have only one item, and we have to put it into a bin, which is incomplete, but there cannot be any other incomplete bins, so the statement is true.

2. Let assume, we have $i = k$ and the statement is true.

3. If we have $i = k+1$, meaning we get another item, then we have 2 cases:

   (a) We can put the new item into an already incomplete bin. Then, we have to investigate 2 sub-cases:

      i. If there is not any incomplete bin with top-item having Hanoi property h_i, then we put this item into the first incomplete bin with top-item with Hanoi property less, than h_i. Thus, we have only 1 bin with this Hanoi property on the top after assigning this item, so the statement is true.

      ii. If there is already a bin b with top-item having Hanoi property h_i, then we have to prove that the i^{th} item is assigned to this bin by First-Fit. We show this indirectly. Let suppose, that First-Fit packs this item into a bin b' such that $b' \neq b$. This can happen only if b' is earlier, than b, but in this case, the top-item of b would have been assigned to b', which is a contradiction. So the statement is true.

   (b) We cannot put the new item into any of the incomplete bins because of the Hanoi conflicts. In this case, there is not any bin with top-item having Hanoi property h_i, so, when we open a new bin to assign the i^{th} item, then this will be the only bin with such top-item. So the statement is true in this case, too.

□

We note, that, although it might looks like Theorem 18 is true for the optimal solution, this is not the case. This is shown by the following example:

Let $I = [1, 2, 2, 4, 4, 4, 3, 3, 3, 3]$ be the list of input items, where each item is given by its Hanoi property, and the capacity of the bins is 5.

The optimal solution for this instance is:

$$\mathsf{OPT}(I) = \{1, 2, 4, 4, 4\}, \{2, 3, 3, 3, 3\}.$$

However, if we want to keep true for every step, that there are no two incomplete bins having top-items with the same Hanoi property, then we get the following:

$$A(I) = \{1, 2, 2, 4, 4\}, \{4\}, \{3, 3, 3, 3\}.$$

Now, using this theorem for First-Fit, we can introduce our main result about the asymptotical approximation ratio of this algorithm.

Theorem 19 *First-Fit algorithm is 1-approximation asymptotically for the BPPHCU model.*

Proof. The proof is based on the idea that First-Fit algorithm uses at most $\mathsf{OPT}(I) + m$ bins, where $\mathsf{OPT}(I)$ is the number of used bins in the optimal solution and m is the number of Hanoi classes. Consequently, we get that $\lim_{|I|\to\infty} \frac{\mathsf{OPT}(I)+m}{\mathsf{OPT}(I)} = 1$. To prove this, we have to consider only the incomplete bins, because these are the ones making difference to the optimum, as even the optimum cannot put more items to the complete bins. As we have seen in Theorem 18, there are no two incomplete bins with top-item having the same Hanoi property, which implies that there are at most as many incomplete bins as Hanoi classes, that is m. Thus, we get that $\mathsf{FF}(I) - \mathsf{OPT}(I) \leq m$, which implies the statement. □

5.2.2 Offline case

As we already mentioned, there are some practical applications, when, despite the fact that all the items are known in advance, we cannot sort them arbitrarily, for example because of lack of space or other resources. In this part, we investigate this case: we have a full list of unit size items with Hanoi conflicts, and we want to assign them into minimum number of bins. We propose an offline algorithm giving optimal solution and we examine its complexity.

In the BPPHCU model, we have finite number of Hanoi properties, thus we can define a finite number of patterns representing the possible loads of the bins. Each pattern p looks like the following:

$$p = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,c} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,c} \end{bmatrix}.$$

Here $p_{i,j}$ is the number of the used bins for the bin-pattern with load j and top-item with Hanoi property i. This implies, that there are $n^{c \cdot m}$ different possible patterns, where n is the number of input items, c is the capacity of each bin and m is the number of Hanoi classes, because there are $c \cdot m$ different bin-patterns, and we can have n used bins from any of them.

Considering these patterns, we can introduce an algorithm, giving optimal solution using a dynamic programming approach represented by the Algorithm 2. Before explaining this method, we have to introduce some notations.

Let P be the set of possible patterns, that is $P \subseteq N^{H\times\{1,2,\ldots,c\}}$. We will generate a matrix $A \subseteq \mathcal{P}(P)^{P\times H}$, giving the set of possible patterns reached from a specific pattern through a transition generated by an item with a specific Hanoi property. This means, that $A[p,h]$ is the set of patterns that can be reached from the pattern p by assigning an item with Hanoi property h to a bin. Furthermore, we will handle a set R_i of possible patterns for each item i in the input, that is $R_i \subseteq \mathcal{P}(P)$.

Now, with these notations we can describe our algorithm finding optimal solution for the sort-restricted BPPHCE model. Firstly, we open the first bin for the first item by adding a pattern to R_1. The added pattern p_{start} is defined as follows:

$$p_{start_{i,j}} = \begin{cases} 1 & \text{if } i = h_1 \text{ and } j = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 2: Dynamic programming algorithm solving the BPPHCU model for fixed order

```
1  R_1.add(p_start) ;                    // initialize with the first item
2  for i := 1 to n − 1 do                    // iterate through the items
3  |  foreach p ∈ R_i do
4  |  |  foreach q ∈ A[p, h_{i+1}] do
5  |  |  |  R_{i+1}.add(q);
6  |  |  |  S[i + 1, q] := p;
7  |  |  end
8  |  end
9  end

   // find the optimum;
10 min := inf;
11 min_p := nil;
12 foreach p ∈ R_n do
13 |  x := sum_{i∈H, j∈{1,2,···,c}} p_{i,j};
14 |  if x < min then
15 |  |  min := x;
16 |  |  min_p := p;
17 |  end
18 end
```

Then, we iterate through the items, and meanwhile we add the possible patterns to R_{i+1}. This is done such that for every item i the elements of $A[p, h_{i+1}]$, that is the possible patterns reached from the pattern p through a transition generated by the item $i+1$, are inserted. Furthermore, the algorithm creates backward pointers for every added pattern to help determine the assignments for the optimal solution. This is handled by the matrix $S \subseteq P^{N \times P}$ containing a pointer to source pattern in R_i.

At the end of this loop, the set R_n contains the possible patterns after every item is assigned, so we only have to count the used items in each patterns and choose the minimal one. The optimal assignment of items into bins can be retrieved by following the pointers of the matrix S.

To determine the running time of this algorithm, we have to find out the sizes of the sets in the matrix A. As each item i can be assigned to any bin having a top-item with Hanoi property less or equal than h_i, and the elements of matrix A contains exactly these successors, we get that for any pattern p and Hanoi property h, $|A[p, h]| = c \cdot h$, because there are c different loads for every top-item, which means, that $c \cdot h$ different patterns can be reached from p through a transition generated by an item with Hanoi property of h. As $h \leq m$, we can say that $c \cdot m$ is an upper-bound on the sizes of the sets in the matrix. Then, we have to specify the size of the R_i sets for every item i, which is clearly upper-bounded by the number of all possible patterns, that is $n^{c \cdot m}$, as shown earlier. So the time complexity of the algorithm is $O\left(n^{c \cdot m+1} \cdot c \cdot m\right)$.

We can see that in this time complexity only c and m occur in exponent, meaning that the running time of the algorithm is polynomial, if we consider c, the capacity of the bins, and m, that is the number of Hanoi classes, as constants, meaning these are independent from the input. This consideration is not uncommon in practical applications, because in several fields fix Hanoi classes are used, such as fragile, or heavy products, so the number of these classes is really independent from the current input items, and the capacity of the containers are also often fix.

6 Conclusion

As conclusion, we can say that we have found some practical applications of the Bin Packing Problem that are not fully handled by the already existing models, so we introduced two relevant variants: the Bin Packing Problem with Hanoi Conflicts (BPPHC) and the Bin Packing Problem with Directed Conflicts (BPPDC).

We deeply investigated the connection of the new models to the Bin Packing Problem with Conflicts (BPPC), which exists in the literature. We pointed out some important connections depending on the type on the conflict graphs. We also examined the importance of the order of the input items. We found that our BPPDC model is the most general one.

Furthermore, we presented results about the complexity of the problems if all the items have unit size. We showed that in this case every Any-Fit algorithm gives not better than $\frac{3}{2}$-approximation for the online version of the Hanoi Conflicts model even for only 2 Hanoi classes, if re-ordering the input is forbidden for the optimum. This lower bound is also generalized for arbitrary number of Hanoi classes. However, we proved that the First-Fit algorithm is asymptotically 1-approximation for this case.

Last, but not least, we proposed an offline algorithm giving optimal solution for the sort-restricted Hanoi Conflicts model with unit size items which has polynomial time complexity, if the capacity and the number of the Hanoi classes are considered as constants.

We believe this work introduced practically important models for the Bin Packing Problem, and further research will be useful to help solving industrial problems.

Acknowledgements

I am very grateful to Csanád Imreh for his sustained help and the useful comments.

References

[1] J. Balogh, J. Békési, Gy. Dósa, L. Epstein, H. Kellerer, Zs. Tuza, Online results for black and white bin packing, *Theory Comput. Syst.*, **56,** 1 (2015) 137–155. ⇒34

[2] N. Bansal, Z. Liu, A. Sankar, Bin-packing with fragile objects and frequency allocation in cellular networks, *Wireless Networks*, **15,** 6 (2009) 821–830. ⇒33

[3] M. Böhm, J. Sgall, P. Vesely, Online colored bin packing, *arXiv:1404.5548 [cs.DS]* (2014) ⇒34

[4] F. Clautiaux, M. Dell'Amico, M. Iori, A. Khanafer, Lower and upper bounds for the Bin Packing Problem with Fragile Objects, *Discrete Appl. Math.*, **163,** 1 (2014) 73–86. ⇒33

[5] Jr., E. G. Coffman, C. Courcoubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber, M. Yannakakis, Bin packing with discrete item sizes part I: Perfect packing theorems and the average case behavior of optimal packings, *SIAM J. Discrete Math.*, **13,** 3 (2000) 384–402 ⇒45

[6] Jr., E. G. Coffman, D. S. Johnson, L. A. McGeoch, R. R. Weber, Bin packing with discrete item sizes part II: tight bounds on First Fit, *Random Structures Algorithms*, **10,** 1–2 (1997) 69–101. ⇒45

[7] Gy. Dósa, L. Epstein, Colorful bin packing, *Algorithm Theory – SWAT 2014*, *Lecture Notes in Comput. Sci.*, **8503** (2014) 170–181. ⇒34

[8] Gy. Dósa, J. Sgall, First Fit bin packing: a tight analysis, *30th International Symposium on Theoretical Aspects of Computer Science: STACS*, Dagstuhl, Germany, 2013, pp. 538–549. ⇒47

[9] Gy. Dósa, Zs. Tuza, D. Ye, Bin packing with 'Largest In Bottom' constraint: tighter bounds and generalizations, *J. Comb. Optim.*, **26,** 3 (2013) 416–436. ⇒34

[10] L. Epstein, On online bin packing with LIB Constraints, *Naval Res. Logist.*, **56,** 8 (2009) 780–786. ⇒34

[11] L. Epstein, Cs. Imreh, A. Levin, Class constrained bin packing revisited, *Theoret. Comput. Sci.*, **411,** 34–36 (2010) 3073–3089. ⇒33

[12] L. Epstein, A. Levin, On bin packing with conflicts, *Approximation and Online Algorithms*, *Lecture Notes in Comput. Sci.* **4368** (2007) 160–731. ⇒34

[13] K. Jansen, An approximation scheme for bin packing with conflicts, *J. Comb. Optim.*, **3,** 4 (1999) 363–377. ⇒34

[14] K. Jansen, S. Öhring, Approximation algorithms for time constrained scheduling, *Inform. and Comput.*, **132,** 2 (1997) 85–108. ⇒34

[15] A. Khanafer, F. Clautiaux, E. G. Talbi, Tree-decomposition based heuristics for the two-dimensional bin packing problem with conflicts, *Computers and Operations Research*, **39,** 1 (2012) 54–63. ⇒34

[16] P. Manyem, Uniform sized bin packing and covering: Online version, *Topics in industrial mathematics*, Springer US, 2000. ⇒34

[17] P. Manyem, R. L. Salt, M. S. Visser, Approximation lower bounds in online LIB bin packing and covering, *J. Autom. Lang. Comb.*, **8,** 4 (2003) 663–674 ⇒34

[18] S. Martello, P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, 1990. ⇒36

[19] B. McCloskey, A. Shankar, Approaches to bin packing with clique-graph conflicts, *EECS Department, University of California, Berkeley* (2005) ⇒34

[20] R. Sadykov, F. Vanderbeck, Bin packing with conflicts: a generic branch-and-price algorithm, *INFORMS J. Comput.*, **25,** 2 (2013) 244–255. ⇒34

[21] H. Shachnai, T. Tamir, Tight bounds for online class-constrained packing, *Theoret. Comput. Sci.*, **321,** 1 (2004) 103–123. ⇒33, 45

[22] H. Shachnai, T. Tamir, Polynomial time approximation schemes for class-constrained packing problems, *Journal of Scheduling*, **4,** 6 (2001) 313–338. ⇒33

[23] H. Shachnai, T. Tamir, On two class-constrained versions of the multiple knapsack problem, *Algorithmica*, **29,** 3 (2001) 442–467. ⇒33

[24] K. Thulasiraman, M. N. S. Swamy, *5.7 Acyclic Directed Graphs*, *Graphs: Theory and Algorithms*, John Wiley and Sons, 1992. 118. ⇒43

[25] J. D. Ullman, The performance of a memory allocation algorithm., *Princeton University, Department of Electrical Engineering, Computer Science Laboratory*, (1971) ⇒47

[26] E. C. Xavier, F. K. Miyazawa, The class constrained bin packing problem with applications to video-on-demand, *Theoret. Comput. Sci.*, **393,** 1–3 (2008) 240–259. ⇒33

Received: January 24, 2015 • Revised: May 2, 2015