Temporal Bin Packing Problems with Placement Constraints: MIP-Models and Complexity

P. Borisovsky[1][0000−0003−2673−7644], A. Eremeev[1][0000−0001−5289−7874], A. Panin[1][0000−0002−1844−6276], and M. Sakhno[1][0000−0002−1844−6276]

Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia
pborisovsky@ofim.oscsbras.ru
eremeev@ofim.oscsbras.ru
aapanin1988@gmail.com
sakhno@ofim.oscsbras.ru

Abstract. In this paper, we investigate new problem statements, generalizing the Temporal Bin Packing Problem (TBPP) with possible applications in cloud computing. We suppose that items are organized into *batches*. All items in the same batch are placed simultaneously. In cloud computing, items correspond to virtual machines (VMs) and batches correspond to user requests for VM placement. In addition, cloud users can create *placement groups* consisting of VMs united by a single placement constraint named *cluster*: at any moment in time, VMs from the same placement group must be hosted on the same rack of servers. In this paper, we consider servers as one-dimensional bins.
We investigate the computational complexity and inapproximability of different formulations of the TBPP with cluster placement constraint and suggest mixed integer programming models for them.

Keywords: Temporal Bin Packing · Placement groups · Virtual machines · Complexity · Inapproximability · MIP model.

1 Introduction

In the classical Bin Packing Problem (BPP) [11], a finite set of items should be placed into the minimum number of identical bins. This problem arises mainly when optimizing container filling and usage. A natural and relevant generalization of BPP is the Temporal Bin Packing problem (TBPP) [2, 5, 6, 15, 17], which is directly related to cloud computing where virtual machines (VMs) are placed on servers. In cloud computing, decisions are made online. Each VM or item is assigned a time interval. The left boundary of the interval represents the time when it is placed or packed, and the right boundary represents the time when it is deleted. The objective is to determine the minimum number of servers (bins) required to host VMs throughout the entire planning horizon.

We investigate new problem statements generalizing TBPP and arising primarily in cloud computing. A cloud has a hierarchical structure that basically consists of regions, network zones, racks and servers. Note that in transportation

logistics a similar problem is known as a Multi-Level Bin Packing Problem [4]. For simplicity, in this study we consider the two-level hierarchy in which sets of servers (bins) are united into racks. Items are organized into *batches* that correspond to user requests for virtual machine placement. A user request occurs at one time, so all virtual machines in the same request are placed at once. In addition, users can create *placement groups* consisting of virtual machines united by a single *placement rule (constraint)* that reflects the dependency between VMs. For example, certain types of applications such as high performance scientific computing or big data processing have to be run on several VMs and produce a large traffic between them and so require a high network bandwidth (such a dependency is known as *affinity*) [3, 13, 18]. To reduce the latency, such VMs are considered as a placement group, and at any moment in time, virtual machines from the same placement group must be hosted on the same network domain or even on the same rack. According to [14], this type of constraints is referred to as *cluster*. Note that there are other forms of dependencies, for example if some data has to be replicated in order to provide a fault tolerance then the corresponding VMs must be placed on different network domains, but in this papers, we concentrate only on the cluster constraints. In our problem settings, input consists of batches of items. These items belong to groups that have the cluster type constraint at the rack level. Each rack consists of a given number of bins, see Fig. 1.

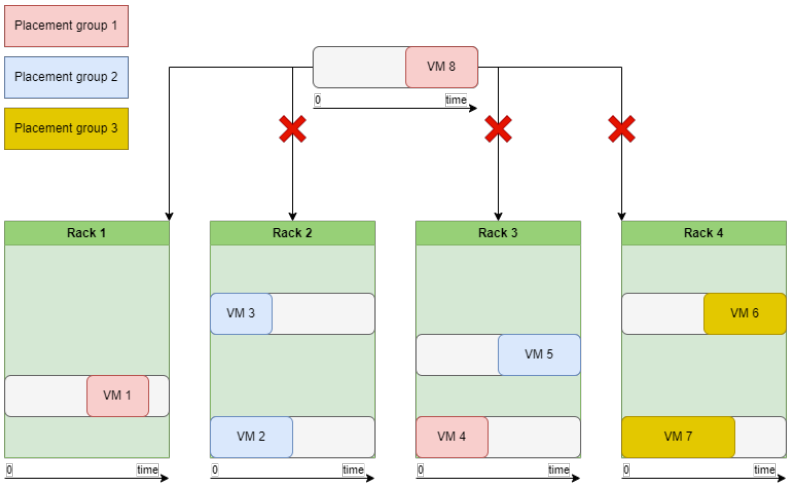

Fig. 1. Illustration of a feasible solution with respect to the cluster placement constraint. Here the lifetime of each VM is shown as an interval on time axis. Let the placement groups be PG1={VM1,VM4}, PG2={VM2,VM3,VM5}, and PG3={VM6,VM7}. Suppose VM1 and VM8 intersect in time. The crosses mark the racks where VM8 can not be placed due to the cluster placement constraint, because it must be put together with VM1.

A common approach to handle a large number of technical requirements and preferences consists in development a multi-objective model, in which among the minimization of active servers there are such additional criteria as power consumption, resource wastage, network traffic, number of rejections and migrations of virtual machines etc. [10, 16, 20]. Comprehensive surveys that cover these and other technological and economical issues of virtual machines scheduling can be found in [9, 12, 19]. In order to keep the models simple and to investigate certain aspects individually, we restrict ourselves to the two VM placement scenarios. In the first one, the set of virtual machines is given and the number of required servers to pack all of them is to be minimized. In the second scenario, there are a fixed number of servers and a sequence of requests, which must be fulfilled without skips until the first placement failure. The goal is to maximize the number of satisfied requests. This formulation can be useful for estimation of the time moment when the datacenter needs to be expanded.

The main purpose of this study is to investigate the complexity of the problem under different settings. Since the basic VM packing problem is a generalization of the classic BPP it is clearly NP-hard. However, there could be particular cases, in which this reasoning may be incorrect. For example, BPP considers items of arbitrary sizes, but the cloud providers usually offer a limited set of VM types, and it is proven in [8] that BPP with the limited number of item types is polynomially solvable. We show that in presence of cluster constraints the problem is NP-hard even for only one VM type. Besides, we consider other practically interesting cases and prove that under certain settings the solutions cannot be approximated with any ratio in polynomial time.

The paper is structured as follows. Section 2 contains detailed description of problems under consideration, together with Mixed Integer Programming (MIP) models for their solution. Section 3 shows different hardness and inapproximability results for formulations from Section 2. Concluding remarks are given in Section 4.

2 Problem Statements and MIP Formulations

Let a set of items $\mathbf{I} = \{1, \dots, I\}$ and rack capacity C (a number of bins in one rack) be given. For each item i, the time interval $[s_i, f_i)$ is known. At the time s_i, the item i must be placed into one of the bins and remain there until the time f_i. A set of event moments $\mathbf{T}$ can be defined as the combination of all left endpoints of time intervals. We can assume $\mathbf{T} = \{1, \dots, T\}$ without loss of generality. Let us denote the weight of item i by w_i and the bin capacity by W.

The items are divided into groups. $\mathbf{G} = \{0, \dots, G\}$ is a set of groups of items. The group $g = 0$ comprises all items without cluster placement constraints. For each group $g \geq 1$ and each event (or time) moment t, all items of g that exist in this moment must be placed on the same rack.

2.1 Problem Statements of Minimizing the Number of Identical Bins

Consider a problem formulation that requires all items to be placed in a minimum number of identical bins, subject to placement constraints. Let $\mathbf{R} = \{1, \ldots, R\}$ be a sufficient set of racks to allocate all items (e.g., $R = [I/C]$). Each rack contains exactly C bins. Let $\mathbf{B}_r$ be a set of bins on a rack r. In addition, we define the following auxiliary sets:
$\mathbf{I}_t = \{i \in \mathbf{I} | s_i \leq t < f_i\}$, where $t \in \mathbf{T}$;
$\mathbf{I}_{gt} = \{i \in \mathbf{I} | s_i \leq t < f_i$ and i belongs to $g\}$, where $t \in \mathbf{T}$ and $g \in \mathbf{G}$.

A Boolean variable z_{rb} takes the value 1 if and only if the bin b of the rack r is used to place items. A Boolean variable x_{rbi} is equal to 1 if and only if the item i is in the bin b of the rack r. To ensure that the placement constraints are met, a Boolean variable a_{rgt} is introduced. The variable takes the value 1 if and only if the items of group g that exist at time t are placed in the rack r.

The problem of minimizing the number of bins can be written in terms of mixed integer linear programming as follows.

$$\sum_{r \in \mathbf{R}} \sum_{b \in \mathbf{B}_r} z_{rb} \to \min_{x,z,a} \tag{1}$$

$$\sum_{r \in \mathbf{R}} \sum_{b \in \mathbf{B}_r} x_{rbi} = 1, \quad i \in \mathbf{I} \tag{2}$$

$$\sum_{i \in \mathbf{I}_t} w_i x_{rbi} \leq W, \quad r \in \mathbf{R}; b \in \mathbf{B}_r; t \in \mathbf{T} \tag{3}$$

$$\sum_{i \in \mathbf{I}} x_{rbi} \leq I \times z_{rb}, \quad r \in \mathbf{R}; b \in \mathbf{B}_r \tag{4}$$

$$\sum_{b \in \mathbf{B}_r} \sum_{i \in \mathbf{I}_{gt}} x_{rbi} = |\mathbf{I}_{gt}| \times a_{rgt}, \quad r \in \mathbf{R}; g \in \mathbf{G} \setminus 0; t \in \mathbf{T} \tag{5}$$

Objective function (1) minimizes the number of bins used. Constraint (2) ensures that all items are placed. Constraint (3) is a limit on the bin capacity. Constraint (4) requires placing items only on used bins. Items from the same group must be on the same rack. This follows from constraint (5).

In cloud computing, it is sometimes required to minimize the number of racks. In this case, it is sufficient to replace the variable z_{rb} with a Boolean variable z_r which takes the value 1 if and only if the rack r is used to place items. Then the objective function (1) and constraint (4) are rewritten as follows:

$$\sum_{r \in \mathbf{R}} z_r \to \min_{x,z,a}$$

$$\sum_{b \in \mathbf{B}_r} \sum_{i \in \mathbf{I}} x_{rbi} \leq I \times z_r, \quad r \in \mathbf{R}$$

2.2 Problem Statement of Maximizing the Number of Batches

Unlike the previous problem, the batch maximization problem aggregates items from the same left endpoint of the time interval into batches. Let a set of batches $\mathbf{U} = \{1, \ldots, U\}$ be given. In fact, in the problem statement under study, the set of batches is equal to $\mathbf{T}$. A Boolean matrix γ determines whether an item belongs to a batch. The value of γ_{ui} is 1 if and only if the item i belongs to the batch u. The Boolean variable y_u determines whether items from the batch u are packed or not.

The batch maximization problem can be written in terms of mixed integer linear programming as follows.

$$\sum_{u \in \mathbf{U}} y_u \to \max_{x,y,a} \tag{6}$$

$$y_{u-1} \geq y_u, \quad 2 \leq u \leq U \tag{7}$$

$$\gamma_{ui} \times \sum_{r \in \mathbf{R}} \sum_{b \in \mathbf{B}_r} x_{rbi} = y_u, \quad u \in \mathbf{U}; i \in \mathbf{I} \tag{8}$$

$$\sum_{i \in \mathbf{I}_t} w_i x_{rbi} \leq W, \quad r \in \mathbf{R}; b \in \mathbf{B}_r; t \in \mathbf{T} \tag{9}$$

$$\sum_{b \in \mathbf{B}_r} \sum_{i \in \mathbf{I}_{gt}} x_{rbi} \leq |\mathbf{I}_{gt}| \times a_{rgt}, \quad r \in \mathbf{R}; g \in \mathbf{G} \setminus 0; t \in \mathbf{T} \tag{10}$$

Objective function (6) maximizes the number of batches packed. Constraint (7) implies that batches are placed sequentially until the first placement failure. Constraint (8) ensures that only whole batches are packed. It is not possible to place only a part of a batch. Constraints (9) and (10) are similar to constraints (3) and (5), respectively.

2.3 Allocation Rate Maximization

We have described mathematical models of the problems of minimizing the size of the resource pool (number of bins or racks) and maximizing the number of batches. Cloud computing also considers statements in which allocation rate is maximized.

Definition 1 (Allocation rate). *The allocation rate is defined as follows. For a given set of items* $\mathbf{I}$ *and for each time moment* t*, let* TW_t *be the total weight consumed by the items of* $\mathbf{I}$ *and* TW *be the total capacity available in the resource pool. Then the allocation rate is* $f = (\max_t TW_t)/TW$*.*

In the problem (1)–(5), all items must be placed. Therefore, we can clearly determine the time when the maximum allocation rate is reached:

$$\bar{t} = \arg\max_t \sum_{i \in \mathbf{I}_t} w_i.$$

The total weight $TW_{\bar{t}}$ is equal to $\sum_{i \in \mathbf{I}_{\bar{t}}} w_i$. The statement of maximizing allocation rate is obtained when replacing the objective function (1) by

$$TW_{\bar{t}} \quad / \quad \sum_{r \in \mathbf{R}} \sum_{b \in \mathbf{B}_r} W \times z_{rb} \to \max_{x,z,a},$$

for the bins minimization case, and by

$$TW_{\bar{t}} \quad / \quad \sum_{r \in \mathbf{R}} W \times C \times z_r \to \max_{x,z,a},$$

for the racks minimization case

For the problem (6)–(10), the statement of maximizing allocation rate is obtained when replacing the objective function (6) by

$$\sum_{t \in \mathbf{T}} q_t \times \sum_{i \in \mathbf{I}_t} \sum_{r \in \mathbf{R}} \sum_{b \in \mathbf{B}_r} w_i x_{rbi} \quad / \quad (W \times R \times C) \to \max_{x,y,a,q}$$

$$\sum_{t \in \mathbf{T}} q_t = 1,$$

where the Boolean variable q_t defines the event moment at which the maximum load of bins is reached.

3 Computational Complexity

In cloud computing, items are virtual machines. The users cannot define the configuration (type) of the virtual machine themselves. Instead, they have to choose from a set of configurations offered by the cloud provider. In practice, TBPP with placement constraints has a limited variety of item types. In our study, the type of an item is its weight. This paper explores two cases where the number of item types is a part of the problem input and there is only one item type, i.e., all items have the same weight.

3.1 The Number of Item Types as a Part of the Problem Input

The first statement corresponds to the scenario where items must be evenly distributed among bins. This approach is commonly used in partition problems. The setting of the classical partition problem is given below.

Partition problem. [7]
Input: Finite set X of items, for each $x \in X$ a weight $a_x \in Z^+$.
Question: Whether the set X can be partitioned into two subsets X^1 and X^2 such that $\sum_{x \in X^1} a_x = \sum_{x \in X^2} a_x$?

The presence of placement groups allows to use the Partition problem ideology for complexity evaluation. This leads to the following statement.

Theorem 1. *The problem of finding a feasible solution for the problem (1)–(5) is NP-hard even in the case of one placement group, one event moment, and any rack capacity $C \geq 2$.*

Proof. Consider an arbitrary instance of Partition problem. Let us construct an instance of the problem (1)–(5) with one placement group and one event moment as follows. The set of items I is exactly equivalent to the set of items X and the weight of each item is equal to corresponding a_x. The group $g = 1$ consists of all items. The capacity of the bin is equal to half the total weight of all items. The rack contains exactly two bins.

In a feasible solution of the constructed instance of the problem (1)–(5), the placement group must be allocated entirely to a single rack. Therefore, the set of items must be distributed into two bins, which corresponds to a positive answer in the NP-complete Partition problem. □

Corollary 1. *The problems of finding a feasible solution for the problems (1)–(5) with the rack minimization criterion and the allocation rate maximization criterion are NP-hard even in the case of one placement group and one event moment.*

Let us turn to the hardness of finding approximate solutions with guaranteed approximation ratio ρ, which is also called *the performance ratio* [1]. Let F_x^* be the optimal value of the objective function for input x (initial data) and $F_x(y)$ be the value of the objective function for solution y obtained by an approximate algorithm.

Definition 2. *An approximate polynomial-time algorithm for a minimization problem is called a ρ-approximation algorithm if it finds a solution y such that $F_x(y) \leq \rho F_x^*$ for any input x.*

Definition 3. *An approximate polynomial-time algorithm for a maximization problem is called a ρ-approximation algorithm if it finds a solution y such that $F_x(y) \geq F_x^*/\rho$ for any input x.*

Implicitly the performance ratio ρ may be a function of x and we can write $\rho(x)$ to emphasize this. So, for an arbitrary minimization problem with input x, the inequality $F_x^* \geq \frac{F_x(y)}{\rho(x)}$ holds. In the case of maximization, we have $F_x^* \leq \rho(x) F_x(y)$.

Corollary 2. *For the problem (1)–(5) with criteria of minimizing the number of racks or bins and maximizing the allocation rate, there are no polynomial-time approximation algorithms with any performance ratio, unless $P = NP$.*

Corollary 2 follows from the fact that the approximation algorithm constructs a feasible solution in polynomial time. Therefore, for the problem (1)–(5), there are no algorithms satisfying Definition 2 or 3. The obtained results show that the

presence of placement constraints strongly complicates the classical Bin Packing-type formulations. For the BPP, polynomial-time approximation algorithms with performance ratio 3/2 exist [1].

Theorem 2. *For the problem (6)–(10), there is no ρ-approximation algorithm for any constant ρ unless $P = NP$.*

Proof. Consider a decision problem P, in which there is a single time moment and one batch consisting of n items is to be allocated on the given set of bins of one large rack. No placement constraints are assumed. It is required to decide whether all of the n items can be packed or not. This problem is NP-complete since it is the corresponding decision problem of the BPP.

Assume for simplicity that ρ is integer. Consider the problem Q to maximize the number of batches that represents ρ copies of problem P. Namely, the first copy of the batch occupies only the time moment $t = 1$, similarly the second copy of the batch exists in the time moment $t = 2$, and so on up to the moment $t = \rho$. Since ρ is a constant, the size of Q is bounded by a polynomial in n.

Suppose there is a ρ-approximation algorithm for Q. If an instance of the decision problem P has a positive answer, then all the batches of problem Q can be packed, i.e. the optimal objective value of problem Q is $F^* = \rho$. In this case, the approximation algorithm will find a solution with $F \geq \frac{F^*}{\rho} = 1$, i.e. it will pack the first batch and therefore solve P. On the other hand, if an instance of problem P has the negative answer, then necessarily $F = 0$.

This allows to correctly decide the problem P in polynomial time, which is impossible unless P = NP. □

Now let us consider the optimization problem, where the solution quality is the maximum allocation rate of packed batches until the first placement failure.

Naturally, for a given solution, the allocation rate is calculated over the batches that are packed in this solution. Note that the optimal solution to the problem that maximizes the number of placed batches until the first failure is an optimal solution to the problem of allocation rate maximization, but not vice versa.

Denote the optimal allocation rate by f^*.

Theorem 3. *For the problem (6)–(10) with allocation rate maximization criterion, there is no R-approximation algorithm for any constant number of racks R and any rack capacity $C \geq 2$, unless $P = NP$.*

Proof. Consider a decision problem P, in which there is a single time moment and a set of n items to be allocated on the given set of bins of one rack. Only one cluster placement group with all items is given. It is required to decide whether all of the n items can be packed or not. This problem is NP-hard even if the total demand of all n items equals to the total resource of a rack (i.e. the allocation rate is equal to 1) because Partition problem reduces to it as a special case (see Theorem 1). Let us limit ourselves to considering only such instances of the problem P.

Consider an optimization problem Q that represents R copies of problem P combined into R batches with different lifetime and different cluster groups, assuming that the resource pool consists of R identical racks as in the problem P. The first copy of items from P make up the first batch and the first group, the second copy of items (indexed from $n+1$ to $2n$) makes up the second batch and group, and so on, up to the batch R and group number R with item indices $n(R-1)+1, \ldots, nR$. Items of batch 1 exist at time period from 1 to R, items of batch 2 exist at time period from 2 to R, etc. Items of batch R exist only at time moment R. Since R is a constant, the size of Q is bounded by a polynomial in n.

Suppose there is a R-approximation algorithm for Q. If an instance of the decision problem P has a positive answer, then all the items of problem Q can be packed, i.e. the allocation rate of problem Q is $f^* = 1$. In this case, the R-approximation algorithm will find a solution with an allocation rate $f \geq \frac{f^*}{R} = \frac{1}{R}$, i.e. it will pack at least the first batch. On the other hand, if an instance of problem P has the negative answer, then necessarily $f < \frac{1}{R}$.

This allows to correctly decide the problem P in polynomial time, which is impossible unless P = NP. □

3.2 The Single Item Type Case

Theorems 1-3 above characterize the computational complexity of the problems under study in the case of unlimited number of item types. The following statement describes the complexity in the case of a single type (and also applies to the cases with any number of types upper-bounded by a constant).

The proof of the following statement relies on a decision version of Graph Coloring Problem.

> **K-coloring problem.** [7]
> **Input:** Graph $G = (V, E)$ and integer $K \leq |V|$.
> A coloring of G into K colors is a partition of V into K independent sets in G.
> **Question:** Can we color G with no more than K colors?

Theorem 4. *The problem (6)–(10) is NP-hard, even in the case of one item type, any fixed number of racks $R \geq 3$, any fixed rack capacity.*

Proof. We show that the K-coloring problem with any fixed number of colors K reduces in polynomial time to the problem (6)–(10) with the number of racks $R = K$ and one bin per rack.

Let us match placement groups subject to the cluster constraint with the vertices of the given graph $G = (V, E), |V| = n, |E| = m$. In our reduction, we will match utilized racks with colors in the K-coloring problem. Placing items from a placement group on a rack will indicate coloring of a vertex that corresponds to this group.

Let the bin capacity be $2n$ and the planning horizon $\mathbf{T}$ be $\{1, \ldots, m+1\}$. For each placement group corresponding to vertex i ($i = 1, \ldots, n$), let's add an item with lifetime from 1 to $m+1$ and weight equal to 1. This way there will be at least one item in each placement group. Since this item exists for the entire time period, the other items in this placement group cannot be packed into the other racks. Given the weights of these items, they can be packed into any bins, even may all be packed into the same bin. For each edge $e_\ell = (i, j), \ell = 1, \ldots, m$, let us add to the placement groups i, j, corresponding to the incident vertices, n items of the same type as described above and set their start time to ℓ and the end time to $\ell + 1$. Assume that each item makes up a separate batch, so the total number of batches is $U = nm + n$.

Figure 2 illustrates the reduction. For example, edge $e_1 = (V1, V2)$ corresponds to four VMs in PG 1 and four VMs in PG 2. Since $l = 1$, they have the time interval [1,2), i.e. they are active at time moment $t = 1$.

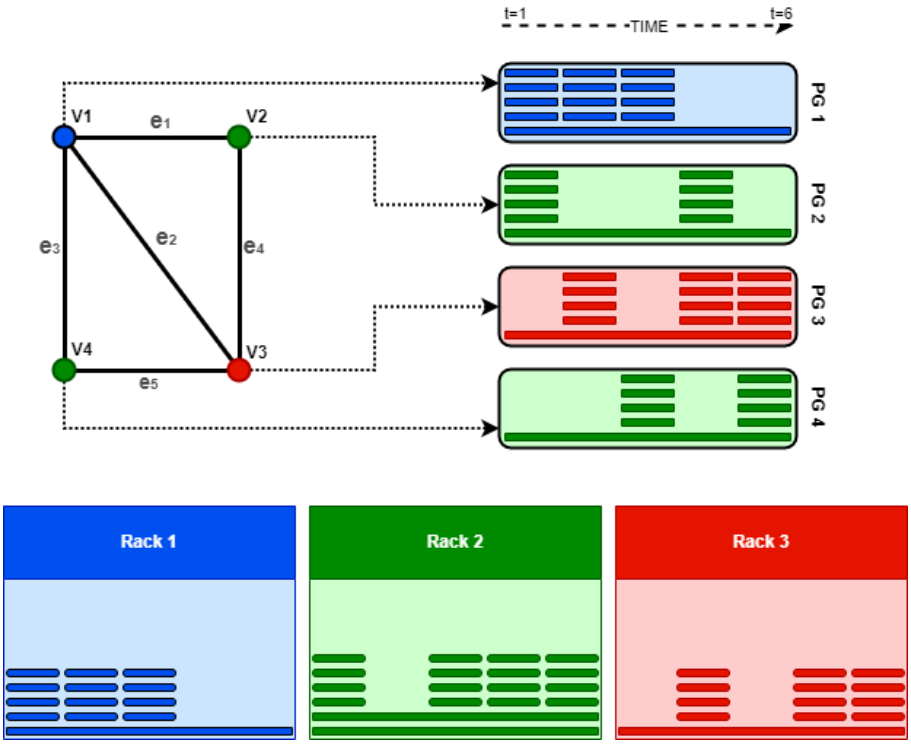

Fig. 2. Illustration of a comparison of instances and solutions of the graph coloring problem and the problems under study in the case of one type of items and unit capacity of racks. A graph with four vertices and five edges is colored in three colors. Each vertex is assigned a placement group. Each color corresponds to a rack.

Note that for any edge $e_\ell = (i, j)$, items from different groups i, j cannot be packed simultaneously on the same rack because together they require $2n + 2$ resource units, while the rack capacity is only $2n$. On the other hand, if two vertices v_i, v_j of the graph can be colored in the same color then groups i, j can be packed into the same rack, because groups i, j have no items living at the same time except for long-lived items, one in each group, since there are no edges between the vertices v_i, v_j in the graph.

The K-coloring problem corresponds one-to-one to the feasible solutions of the constructed instance with $R = K$ racks, and the number of the allocated batches in such solutions is equal to the total number of batches U, i.e. the objective value is equal to U in these solutions. Thus, answering the question: "Is the optimum to the problem (6)–(10) equal to at least U batches?" allows to solve the K-coloring problem. So the latter problem reduces to the problem (6)–(10) with $R = K$ in polynomial time.

Since the reduction yields only instances in which numerical parameters are at most polynomial in n and the K-coloring problem is NP-complete [7] for any fixed number of colors $K \geq 3$, the considered problem is NP-hard. □

Corollary 3. *The problem (1)–(5) is NP-hard with all optimization criteria considered in the paper, even in the case of one item type, any maximum number of racks $R \geq 3$.*

Proof. Placing items in no more than K racks corresponds to a positive answer in the K-coloring problem. □

Corollary 4. *For the problem (6)–(10) even in the case of one item type, any maximum number of racks $R \geq 3$, in the presence of cluster constraints, there is no ρ-approximation algorithm for any constant ρ unless $P = NP$.*

Proof. It is enough to assume that the problem P in the proof of Theorem 2 has cluster constraints and a given number of time moments T (and so it is NP-complete), the problem Q is constructed as ρ copies of P with ρT moments, then the rest of the proof can be reproduced. □

4 Conclusions

In this paper, we have shown that cluster placement constraint significantly complicates the Temporal Bin Packing problem. These are the theoretical results, but our preliminary experiments with modern MIP solvers also indicate that TBPP with placement constraints requires much more CPU time and memory. We expect that future research will provide a detailed experimental analysis of the behaviour of MIP models and algorithms on such problems. Also, we hope that certain special cases will be identified, where TBPP with placement constraints admits constant-factor approximation algorithms.

Acknowledgement

The research was carried out in accordance with the state task of the IM SB RAS (projects FWNF-2022-0020 and FWNF–2022–0019).

References

1. Ausiello, G., Crescenzi, P., Gambosi, G. et al., Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer-Verlag, Berlin (1999)
2. de Cauwer, M., Mehta, D., O'Sullivan, B.: The temporal bin packing problem: An application to workload management in data centres. In: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), San Jose, CA, pp. 157–164. IEEE (2016)
3. Chen, J., He Q., Ye, D., Chen, W. , Xiang, Y., Chiew, K., Zhu, L.: Joint affinity aware grouping and virtual machine placement. Microprocessors and Microsystems **52**, 365–380 (2017)
4. Chen, L., Tong, X., Yuan, M., Zeng, J., Chen, L: A Data-Driven Approach for Multi-level Packing Problems in Manufacturing Industry. In: 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19), Anchorage, USA, pp. 1762–1770. ACM (2019)
5. Dell'Amico, M., Furini, F., Iori, M.: A branch-and-price algorithm for the temporal bin packing problem. Computers & Operations Research. **114**, 104825 (2020)
6. Furini, F., Shen, X.: Matheuristics for the temporal bin packing problem. In: Recent Developments in Metaheuristics, vol. 62, pp. 333–345. Springer (2018)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co, San Francisco, Calif. (1990)
8. Goemans, M.X., Rothvoss, T.: Polynomiality for Bin Packing with a Constant Number of Item Types. Journal of the ACM **67**(6), 1–21 (2020)
9. Grushin, D.A., Kuzyurin, N.N.: On Effective Scheduling in Computing Clusters. Programming and Computer Software **45**(7), 398–404 (2019)
10. Guo, X.: Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. Alexandria Engineering Journal **60**(6), 5603–5609 (2021)
11. Johnson, D. Near-optimal bin packing algorithms. Ph. D. Thesis, Dept. of Mathematics, M.I.T., Cambridge, MA (1973)
12. Mann, Z.: Allocation of Virtual Machines in Cloud Data Centers – A Survey of Problem Models and Optimization Algorithms. ACM Computing Surveys **48**(1), 1–34 (2015)
13. Pachorkar, N., Ingle, R.: Multi-dimensional Affinity Aware VM Placement Algorithm in Cloud Computing. International Journal of Advanced Computer Research **3**(4), 121–125 (2013)
14. Placement groups. (n.d.). Amazon Elastic Compute Cloud. URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html
15. Ratushnyi, A., Kochetov, Y.: A column generation based heuristic for a temporal bin packing problem. In: P. Pardalos, M. Khachay, A. Kazakov (eds). Mathematical Optimization Theory and Operations Research (MOTOR), LNCS, vol. 12755, pp. 96–110. Springer, Cham (2021)

16. Regaieg, R., Koubàa, M., Osei-Opoku, E., Aguili, T.: Multi-objective mixed integer linear programming model for VM placement to minimize resource wastage in a heterogeneous cloud provider data center. In: 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), Prague, Czech Republic, pp. 401–406. IEEE (2018)
17. Sakhno, M.: A grouping genetic algorithm for the temporal vector bin packing problem. In: 19th International Asian School-Seminar on Optimization Problems of Complex Systems (OPCS), pp. 94–99, IEEE (2023)
18. Su, K., Xu, L., Chen,C., Chen W., Wang,Z.: Affinity and Conflict-Aware Placement of Virtual Machines in Heterogeneous Data Centers. In: 2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems, Taichung, Taiwan, pp. 289–294. IEEE (2015)
19. Talebian, H., Gani, A., Sookhak, M. et al.: Optimizing virtual machine placement in IaaS data centers: taxonomy, review and open issues. Cluster Comput. **23**, 837–878 (2020)
20. Zheng, Q., Li, R., Li, X., Shah, N., Zhang, J., Tian, F., Chao, K.-M., Li, J.: Virtual machine consolidated placement based on multi-objective biogeography-based optimization. Future Generation Computer Systems **54**, 95-122 (2016)