Annals of Operations Research (2024) 333:481–499
https://doi.org/10.1007/s10479-023-05446-8

ORIGINAL RESEARCH

Worst-case analysis of heuristic approaches for the temporal bin packing problem with fire-ups

John Martinovic[1] · Nico Strasdat[1]

Accepted: 6 June 2023 / Published online: 1 July 2023

Abstract

We consider the temporal bin packing problem with fire-ups (TBPP-FU), a branch of operations research recently introduced in multi-objective cloud computing. In this scenario, any item is equipped with a resource demand and a lifespan meaning that it requires the bin capacity only during that time interval. We then aim at finding a schedule minimizing a weighted sum of the total number of bins required and the number of switch-on processes (so-called fire-ups) caused during operation. So far, research on the TBPP-FU has mainly focused on exact approaches and their improvement by valid cuts or variable reduction techniques. Although these studies have revealed the problem considered here to be very difficult to cope with, theoretical contributions to heuristic solution methods have not yet been presented in the available literature. Hence, in this article we investigate the worst-case behavior of some approximation algorithms, ranging from classic online algorithms to a more sophisticated look-ahead heuristic specifically designed for the TBPP-FU. In addition, we theoretically study three heuristics the ideas of which are inspired by solution methods for generalized bin packing problems in the field of logistics. As a main contribution, we constructively show that the feasible solutions obtained by all these approaches can be arbitrarily bad. By doing so, we (i) identify a new open problem in cutting and packing, and (ii) establish another previously unknown difference between the classical TBPP and the extended problem with fire-ups, rendering the latter the more difficult problem even from a heuristic point of view.

Keywords Cutting and packing · Temporal bin packing · Fire ups · Heuristics · Worst-case analysis

Mathematics Subject Classification 90C59 · 90C10

✉ John Martinovic
john.martinovic@tu-dresden.de

Nico Strasdat
nico.strasdat@tu-dresden.de

1 Institute of Numerical Mathematics, Technische Universität Dresden, Dresden, Germany

1 Introduction

The *temporal bin packing problem (TBPP)* generalizes the classic BPP, see Delorme et al. (2016) and Scheithauer (2018), with respect to an additional time dimension. More precisely, any item $i \in I := \{1, \ldots, n\}$ is specified by a *resource demand* (or *item size*) $c_i \in \mathbb{Z}_+$ that has to be satisfied only during the *lifespan* $[s_i, e_i)$ of that item, where $s_i, e_i \in \mathbb{Z}_+$ with $s_i < e_i$ denote the *starting and ending time*, respectively. A set of given items then has to be assigned to as few bins as possible while respecting the bin capacity $C \in \mathbb{Z}_+$ at any instant of time. It is important to note that even though the relationships to two-dimensional packing problems seem obvious, the TBPP is an independent problem in operations research. This is particularly due to the fact that the bin capacity represents a renewable resource at every instant of time, and, consequently, items do not have to occupy the same units (of the bin) over their entire lifespan, see Dell'Amico et al. (2020) and Martinovic et al. (2021) for a more detailed explanation.

Although the TBPP is a fairly natural extension of the extensively studied BPP, its scientific foundations have been driven mainly by previous research on the *temporal knapsack problem (TKP)*, see Bartlett et al. (2005), Caprara et al. (2013) and Gschwind and Irnich (2017). Consequently, the TBPP was first described rather lately in the relevant literature in an application-oriented publication from the field of computer science, see de Cauwer et al. (2016). Nonetheless, addressing the exact solution of the TBPP has been successfully advanced by two sophisticated approaches, namely a branch-and-bound algorithm (using Ryan-Foster branching together with a wide variety of different bounds), see Dell'Amico et al. (2020), and a layer-based combinatorial arcflow model of manageable exponential size, see Martinovic et al. (2023). Surprisingly, searching the relevant literature for contributions on heuristic methods for the TBPP does not immediately lead to the desired results. This is not because such approaches do not exist at all, but rather because they were already discussed in early publications on the so-called *dynamic bin packing problem* about 40 years ago (and thus well before the introduction of the term TBPP), see Coffman et al. (1983). As a consequence of that, it seems that any follow-up article dealing with that topic, like Chan et al. (2008, 2009), has stuck to this original terminology rather than harmonizing it with the parallely evolving "temporal notations". These publications mainly focussed on rather simple heuristics the properties of which were already well studied for the classical BPP. To be more precise, special emphasis was given to the following iterative online[1] algorithms:

– *any-fit (AF)*, scheduling the current item to an arbitrary open bin, see Chan et al. (2008),
– *first-fit (FF)*, assigning the current item to the lowest-indexed open bin, see Chan et al. (2008, 2009) and Coffman et al. (1983)
– *best-fit (BF)* and *worst-fit (WF)*, trying to pack an item into the open bin with the currently largest (BF) or smallest (WF) load, respectively, see Chan et al. (2008).

For all these heuristics, the quality of the feasible solutions obtained has been studied thoroughly and, in many cases, tight approximation factors could be found. In this context, it is remarkable that none of these publications is cited in the most recent TBPP literature, such as Dell'Amico et al. (2020) and Martinovic et al. (2023), suggesting that the existence of these theoretical contributions is largely unknown to the cutting and packing community. For this reason, we will briefly summarize some of the results in the next section, also to better display the differences that arise when so-called *fire-ups* are included.

[1] An online algorithm has to make its decision just with the information available when placing the current item. In particular, there is no further knowledge of which items will arrive next.

Considering fire-ups in item-to-bin assignments is a relatively new aspect of modelling and optimization introduced as the *temporal bin packing problem with fire-ups (TBPP-FU)* in an application from the field of cloud computing, see Aydin et al. (2020). The basic idea is that an unused server (or bin) can be temporarily put into some idle mode to save energy, see Fettweis et al. (2019), but it has to be re-activated later if necessary.[2] Any such transition from an empty state into active operation is counted as one fire-up and it is rather energy-intense, meaning that, as a second objective, the number of fire-ups should be kept small to operate sustainably. Roughly spoken, a low number of fire-ups relates to continuous operation of the servers or a scenario where servers can be switched off without being required again later.

Typically, a weighted sum method (scaling the number of fire-ups by some parameter $\gamma > 0$) is used to address both goals together in one objective function, see Aydin et al. (2020). More abstractly, using a server induces costs for the pure provision of the server resources, but additionally also "temporal costs" depending on the operation mode of this server (specified by the interaction of all items on the server). Note that the latter is different to, for instance, the cost terms appearing in generalized bin packing applications in the field of logistics, see Baldi et al. (2019) and Crainic et al. (2021), where typically the item-dependent costs (of a schedule) are only influenced by the individual item-to-bin assignment decisions (and not the overall packing pattern). A more detailed discussion of these and other fundamental differences between the temporal problems considered here and those encountered in the previously mentioned application is part of Sect. 4.

Although there is a quite strong relation between the TBPP-FU and the TBPP, research has shown that important properties are lost as a consequence of the extended problem statement. In the literature, the two main differences are given by:

- An optimal solution to the TBPP-FU typically uses more bins than required in an optimal configuration without considering fire-ups, see Example 2.2 in Aydin et al. (2020). Hence, solving the TBPP does not necessarily lead to an upper bound on the number of bins required in the TBPP-FU.
- In general, temporal decompositions cannot be applied to the TBPP-FU, see Theorem 3 in Martinovic and Strasdat (2022), meaning that an instance typically cannot be split into independent subinstances of smaller size.

So, not only the solution sets of the two problems may be completely disjoint, but also some fundamental techniques exploiting the structural properties of an instance cannot be used to obtain these solutions in case of the TBPP-FU, in general. As a consequence of that, research has mainly dealt with improving the ILP formulations (called M1 and M2) proposed in Aydin et al. (2020) by various aspects like symmetry breaking conditions and valid cuts, see Martinovic et al. (2021), as well as clique-based reduction methods or the use of heuristic information, see Martinovic et al. (2022). For the latter, the *constructive look-ahead heuristic (CLH)* introduced in Aydin et al. (2020) was used, since it is the only approximation algorithm, known in the literature, specifically addressing the fire-up term in the objective function. While empirically it was shown that lots of variables and constraints can be removed based on the heuristic solution, its theoretical properties have not been dealt with at all so far.

In this article, we would therefore like to focus on the approximation guarantee of heuristic approaches for the TBPP-FU. As already alluded to earlier, we will start with a short repetition of standard online algorithms known from the classic TBPP, collect their main theoretical properties ($\rightarrow$ Sect. 2), and show that their worst-case performance ratio is no longer bounded,

[2] Note that similar ideas are also discussed in the field of thermal units, see Frangioni and Gentile (2006).

when fire-ups have to be respected ($\rightarrow$ Sect. 3). As a main contribution, we prove the same result for CLH thus closing an open theoretical question for the only TBPP-FU heuristic known in the literature ($\rightarrow$ Sect. 3). Moreover, we also theoretically study further types of constructive heuristics the ideas of which are based on neighboring bin packing applications from the field of logistics ($\rightarrow$ Sect. 4). In total, we prove that eight different heuristics for the TBPP-FU possess an unbounded approximation guarantee. Altogether, our investigations are not only the first to cover theoretical properties of heuristic approaches for the TBPP-FU, but they do also establish a third fundamental difference between the problem under consideration and the underlying TBPP, that is, the hardness of finding reasonably good approximate solutions by (common) heuristics.

2 Heuristics for the TBPP: an overview and important results

Let us start with the following definition:

Definition 1 A tuple $E = (n, C, \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{e})$, where $\boldsymbol{c}, \boldsymbol{s}$, and $\boldsymbol{e}$ are n-dimensional vectors collecting the input-data (item size, starting time, ending time) of the items, is called *an instance* (of the TBPP).

Without loss of generality, we assume the items to be sorted with respect to non-decreasing starting times (breaking ties in an arbitrary way) and to satisfy $c_i \leq C$ to ensure solvability. For any given algorithm ALG, we define the *(worst case) performance ratio* $\sigma := \sigma(ALG)$ by

$$\sigma(ALG) := \sup_E \frac{ALG(E)}{OPT(E)},$$

with $ALG(E)$ and $OPT(E)$ denoting the heuristic and the optimal value (of E), respectively. For a fixed instance E, the optimal value $OPT(E)$ can either be determined by theoretical arguments or by an exact formulation. For the sake of exposition, here we just mention the textbook formulation given in Dell'Amico et al. (2020) as one example. It is based on classic assignment variables $x_{ik} \in \{0, 1\}$, where $x_{ik} = 1$ holds if and only if bin $k \in K$ (with K denoting some index set of the bins) carries item $i \in I$. In addition, there are bin-dependent variables $z_k \in \{0, 1\}$ with $z_k = 1$ if and only if bin k is used. Then, we obtain the

Assignment Model for the TBPP

$$z = \sum_{k \in K} z_k \rightarrow \min$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1, \qquad i \in I, \tag{1}$$

$$\sum_{i \in I_t} c_i x_{ik} \leq C \cdot z_k, \qquad t \in T, k \in K, \tag{2}$$

$$x_{ik} \in \{0, 1\}, \qquad i \in I, k \in K, \tag{3}$$

$$z_k \in \{0, 1\}, \qquad k \in K. \tag{4}$$

The objective function minimizes the total number of servers in use. Moreover, the two sets of constraints make sure that any job is executed precisely once (see (1)) and that the capacity of the servers is respected at any instant of time $t \in T := \bigcup_{i \in I} \{s_i, e_i\}$ (see (2)).

Additionally, the latter prevent jobs from being assigned to unused servers at all. Note that I_t is an index set collecting the items $i \in I$ with $t \in [s_i, e_i)$, i.e., the items active at time $t \in T$. Some improvements of this textbook model as well as further (more sophisticated) exact approaches can be found in Dell'Amico et al. (2020) and Martinovic et al. (2023).

Although there is a certain body of work dealing with heuristics for the TBPP, as mentioned earlier, the fact that they were all published with respect to a completely different terminology might be the reason why there is no link between the most recent literature dealing with exact approaches (partly requiring and benefiting from heuristic information) and the former theoretical results related to what was called dynamic bin packing. To close this gap, let us briefly repeat the most important results obtained at that time.

The first heuristic proposed in the literature is of first-fit type, see Coffman et al. (1983), and an interval for the performance ratio is given by the following result.

Theorem 1 (see Theorem 2 and Theorem 6 in Coffman et al. (1983)) *We have*

$$2.389 \approx \frac{43}{18} \leq \sigma(FF) \leq \frac{5}{2} + \frac{3}{2}\log\left(\frac{\sqrt{13}-1}{2}\right) \approx 2.897.$$

The lower bound also holds for any arbitrary online algorithm.

The proofs related to these bounds are very technical and shall therefore be omitted. Among others, establishing the lower bound requires an instance construction containing eleven steps (partly with several subcases). Note that, even if the true value of $\sigma(FF)$ was not identified in that early publication, the results obtained are nevertheless quite remarkable:

- First of all, we see that adding a temporal dimension to the classical BPP makes it much harder to find a feasible solution of good quality with reasonable numerical efforts. By that, we particularly mean that the known approximation factor of FF for the BPP [that is, 1.7, see Dósa and Sgall (2013)] is possibly raised by up to more than one unit.
- Secondly, already this very first article dealing with heuristics for the TBPP was able to establish a lower bound for a wide variety of approximation algorithms.

In Chan et al. (2008), the authors present new results for all the simple heuristics mentioned in the above list, see Sect. 1, but some of their considerations are limited to *unit fraction* item sizes (meaning that the bin capacity C is an integer multiple of any c_i, $i \in I$). In the cutting and packing literature, such a scenario is sometimes also referred to as the *divisible case*, see Coffman et al. (1987), Marcotte (1983) and Martinovic (2022).

Remark 1 Note that an upper bound for $\sigma(AF)$ also holds for FF, BF, and WF, since the latter are, in a sense, "special cases" that can occur in the random bin selection process of AF.

In particular, the following results are obtained:

Theorem 2 (see Theorem 7 and Theorem 8 in Chan et al. (2008)) *We have* $\sigma(WF) \geq 3$ *and* $\sigma(BF) \geq 3$.

Interestingly, in Theorem 6 in Chan et al. (2008), the tightness of these approximation factors (of BF and WF) for instances with unit fraction item sizes was shown. In fact, the corresponding proof can be easily extended to arbitrary instances, so that even the following result holds:

Theorem 3 *We have* $\sigma(AF) \leq 3$.

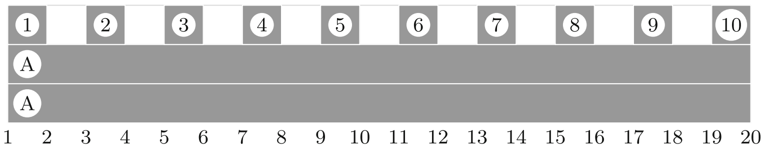

Fig. 1 An illustration of $E(\alpha, \beta)$ for $\alpha = 10$ and $\beta = 0$

In the light of Remark 1, we now know that BF and WF do possess a worst-case performance ratio of 3. This is a remarkable qualitative difference to the situation for the classic BPP due to two reasons:

- Firstly, we see that there is no difference between BF and WF in terms of the approximation guarantee. For the BPP, BF is known to perform better than WF, see Dósa and Sgall (2014), Johnson (1973).
- Secondly, FF is better than BF for the TBPP, whereas from a worst-case perspective both of them were equivalent for the BPP, see Dósa and Sgall (2013), Dósa and Sgall (2014).

As a last point, we mention that for FF the following two improvements of the lower bounds from Coffman et al. (1983) can be obtained:

Theorem 4 (see Theorem 1 and Theorem 5 in Chan et al. (2008)) *We have* $\sigma(FF) \geq 2.45$. *Moreover, for the subclass of unit fraction item sizes the performance ratio of FF is bounded above by a constant less than* 2.5.

Theorem 5 (see Theorem 1 in Chan et al. (2009)) *For any online algorithm, we have* $\sigma(ALG) \geq \frac{5}{2}$.

Both together imply that approximating the TBPP is harder for arbitrary item sizes than for unit fractions. However, the exact performance guarantee of FF is still not known in either case.

3 Heuristics for the TBPP-FU: a worst-case analysis

Let us define a family of TBPP-FU instances $E(\alpha, \beta)$ parametrized by $\alpha, \beta \in \mathbb{Z}_+$ with $\alpha \geq 1$. More precisely, any such instance uses the bin capacity $C = 2$, some scaling parameter $\gamma > 0$, and is given by the following items:

- two items (Type 'A') with $[s_A, e_A) = [1, 2\alpha)$ and $c_A = 1$,
- β items (Type 'B') $[s_B, e_B) = [1, 2\alpha)$ and $c_B = 2$,
- α items (Type 'C', labelled from 1 to α) with $[s_i, e_i) = [2i - 1, 2i)$ and $c_i = 1$, $i = 1, \ldots, \alpha$.

Note that γ does not affect the shape of the items, so that we do not have to specify this value when illustrating an instance. Some exemplary configurations are given in Figs. 1 and 2.

Remark 2 Counterexamples in the field of the TBPP-FU often benefit from the interaction of relatively few long jobs and many very short jobs, see also Example 2.2 in Aydin et al. (2020), where an instance with $C = 4$ and three different item types (either short or long) with $c_i \in \{2, 3\}$ is used to state that an optimal solution to the TBPP-FU does not have to use the minimum number of bins possible. In fact, this is an important result, because it demonstrates the first key difference between the TBPP and the TBPP-FU from a structural (and algorithmic) point of view.

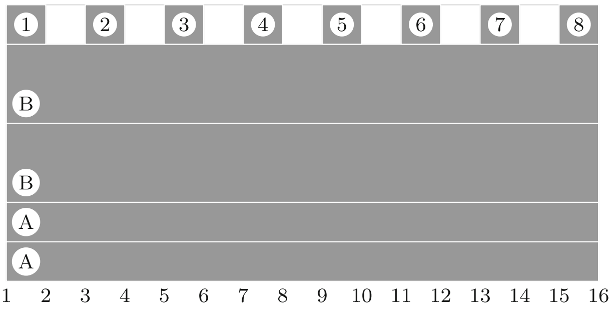

Fig. 2 An illustration of $E(\alpha, \beta)$ for $\alpha = 8$ and $\beta = 2$

With the help of these instances, we will show that any heuristic from the literature proposed for the TBPP and the TBPP-FU can be arbitrarily bad. To this end, let us first construct an optimal solution of $E(\alpha, \beta)$.

Theorem 6 *For any feasible choice of* (α, β) *and any scaling parameter* $\gamma > 0$ *we have*

$$OPT(E(\alpha, \beta) = (1 + \gamma) \cdot (\beta + 2).$$

Proof Given the items available at $t = 1$, at least $\beta + 2$ bins are required in an optimal solution. Hence, it suffices to find a feasible solution using precisely this number of bins in continuous operation (i.e., with one fire-up per bin). To achieve this, we consider the following assignment:

– Any of the β items of type 'B' requires a separate bin, leaving no space for any other item to be added.
– We pack one item of type 'A' together with all the α items of type 'C'.
– The last bin just contains one item of type 'A'.

Altogether, this solution uses $\beta + 2$ bins each having exactly one fire-up (at the very beginning). Hence, the optimal value is given by $(1 + \gamma) \cdot (\beta + 2)$ and the claim is proved. □

As a direct consequence of that, we can state:

Theorem 7 *For any* $\gamma > 0$*, the worst-case performance ratio of AF, FF, BF, and WF (applied to the TBPP-FU) is unbounded.*

Proof Let us consider the instance $E(\alpha, 0)$ with some arbitrary $\gamma > 0$, see also Fig. 1. Then, in any of the heuristics mentioned before the items are placed in the same way since there is precisely one possibility in every iteration. Hence, either way, we end up with one bin grouping the two items of type 'A' (that is, one fire-up) and one bin collecting all items of type 'C' (that is, α fire-ups). Altogether, we have $ALG(E(\alpha, 0)) = 2 + \gamma \cdot (1 + \alpha)$, meaning that the ratio $ALG(E)/OPT(E)$ is unbounded when α tends to infinity. □

Remark 3 Of course, the heuristics studied so far are not tailored to tackle the TBPP-FU since only a part of the objective function (namely, the number of bins) is addressed in the iterative decisions. However, in general, this fact alone is not sufficient to explain the resulting change towards an unbounded approximation guarantee. For example, the same FF algorithm applied to the problem of *busy time minimization with bounded parallelism* (i.e., an objective function that no longer includes the number of bins as a cost term at all) would still yield a 4-approximation, see Flammini et al. (2010).

Remarkably, the previous theorem also holds for any other iterative online algorithm, since the reason for the bad performance is that the algorithm is not allowed to

(I) open an additional bin when the existing bins are able to accommodate the currently considered item,
(II) consider several items (and their expected interaction) at once and place them together on the most suitable server.

In fact, these restrictions cannot be relaxed either, because an online algorithm does not know which items will follow in future (if any), and so – by way of example – there is no basis for deciding whether to add an extra bin or not.

Although these critical features have not been identified or reported before, the constructive look-ahead heuristic (CLH) proposed in Aydin et al. (2020), see Algorithm 1, was intuitively equipped with some tailored modifications addressing issue (I). First of all, it is not an online algorithm since it uses a certain amount of future items (specified by q) when making the current decision. Moreover, and even more importantly, it allows to open an additional bin at every stage to possibly get around the critical situation observed in the previous example. Indeed, CLH would solve the instance displayed in Fig. 1 and any other instance $E(\alpha, 0)$ correctly. Note that heuristics focussing on issue (II) from the above list will be discussed later in Sect. 4.

Algorithm 1 CLH with look-ahead parameter q

```
Input: Item list ordered by non-decreasing starting times $s_i$, parameter $q \in \mathbb{N}$.
1: Initialize the "empty" assignment $\mathcal{A}(0) := \emptyset$.
2: for $i \in I$ do
3:     Assign item $i$ to any open bin of $\mathcal{A}(i-1)$ that can accommodate it and (as another alternative) also
       to a new empty bin. By that we obtain the assignments $\mathcal{A}_1 := \mathcal{A}_1(i), \ldots, \mathcal{A}_p := \mathcal{A}_p(i)$ for some
       $p := p(i) \leq n$.
4:     Add the next $q$ items (or less, if $i + q > n$) to each of these allocations in a best-fit fashion and obtain
       the (updated) assignments $\widetilde{\mathcal{A}}_1, \ldots, \widetilde{\mathcal{A}}_p$.
5:     Choose one assignment from $\widetilde{\mathcal{A}}_1, \ldots, \widetilde{\mathcal{A}}_p$ with the lowest objective value (breaking ties with the lower
       index $s \in \{1, \ldots, p\}$ of the assignment), say $\widetilde{\mathcal{A}}_{opt}$, and define $\mathcal{A}_{opt}$ as $\mathcal{A}(i)$, that is, the starting point
       of the next iteration.
6: end for
Output: heuristic solution with objective value $CLH_q(E)$.
```

Remark 4 Note that a deeper look into the future (i.e., a larger value of q) does not necessarily lead to an equivalent or even better result. More formally, the property $CLH_q(E) \geq CLH_{q+1}(E)$ does not hold, in general. By way of example, let us consider the instance E defined by $n = 5$, $C = 2$, $\gamma = 1$, and the following item characteristics

$$\boldsymbol{c} = (1, 1, 1, 1, 2), \quad \boldsymbol{s} = (1, 1, 2, 4, 5), \quad \boldsymbol{e} = (6, 6, 3, 5, 6).$$

For $q = 1$, CLH finds an optimal solution using two bins and three fire-ups by forming the bins $B_1 = \{1, 2\}$ and $B_2 = \{3, 4, 5\}$. Contrary to that, $q = 2$ leads to a worse objective value by requiring the three bins $\widetilde{B}_1 = \{1, 3, 4\}$, $\widetilde{B}_2 = \{2\}$, and $\widetilde{B}_3 = \{5\}$, together with three fire-ups. Hence, there is no strict relation between q and the objective value obtained by Algorithm 1.

By inserting sufficiently many items of type 'B', as in Fig. 2, the possibilities to successfully look into the future can be limited, especially for the crucial decision how to deal with the first two items (of type 'A'). Hence, for the instance family denoted by $E(\alpha, \beta)$, we obtain the following result, in which we refer to Algorithm 1 and a fixed choice of q by CLH_q.

Theorem 8 *For any feasible choice of* $(\alpha, \beta) \in \mathbb{N} \times \mathbb{Z}_+$, $\gamma > 0$, *and* $q \in [0, \beta + 1 + \alpha]$ *we have*

$$CLH_q(E(\alpha, \beta)) = \begin{cases} \beta + 2 + \gamma \cdot (\beta + 1 + \alpha), & \textit{if } q \leq \beta + 1, \\ (1 + \gamma) \cdot (\beta + 2), & \textit{if } q \geq \beta + 2. \end{cases}$$

Proof Let $E(\alpha, \beta)$ be an instance constructed according to the rules stated at the beginning of Sect. 3. First of all, note that the maximum lookahead for item $i = 1$ in Algorithm 1 is equal to $\beta + 1 + \alpha$, given the total number of items in the considered instance. Now let us go through the two cases:

(I) For $q \leq \beta + 1$, we have the following observations according to the rules given in Algorithm 1:

- The first item of type 'A' is scheduled to bin $k = 1$.
- The second item of type 'A' can either enter bin $k = 1$ (Assignment $\mathcal{A}_1$) or open a new bin $k = 2$ (Assignment $\mathcal{A}_2$). The decision between these two possibilities is done by involving the next $q \leq \beta + 1$ items in a best-fit fashion. For $q \leq \beta$, the next q items are all of type 'B', and so they will consume precisely one bin in any feasible assignment. Hence, in terms of the objective value, CLH will choose to pack both items of type 'A' into the same bin (that is, $k = 1$). Technically, for $q = \beta + 1$ there is a tie between
  - assigning the items of type 'A' together in the same bin $k = 1$, any item of type 'B' in a separate bin, and to open a new bin $k = \beta + 2$ for item $i = 1$
  - assigning the items of type 'A' to two different bins ($k = 1$ and $k = 2$), any item of type 'B' in a separate bin, and to add item $i = 1$ to (the lower-indexed) bin $k = 1$.

  Given the rules of CLH, the candidate assignment with the lower index (that is, assignment $\mathcal{A}_1$) is chosen in such a situation. Hence, we end up with the same configuration for any $q \leq \beta + 1$.
- The next β items (all type 'B' items) will always be assigned to a separate bin, since they cannot be packed in any of the existing bins. Due to the same reason, the first item of type 'C' (labelled $i = 1$) has to open a new bin (with index $k = \beta + 2$).
- The remaining items of type 'C' can either go to the same bin (as $i = 1$) or open an additional one. However, the latter will lead to a larger objective value for any $\gamma > 0$. Hence, they are all scheduled to the same bin $k = \beta + 2$.

Altogether, we have $\beta + 1$ bins with precisely one fire-up and one bin with α fire-ups, summing up to $\beta + 2$ bins and $\beta + 1 + \alpha$ fire-ups. This proves the the first part of the claim.

(II) For $q \geq \beta + 2$, we have the following observations according to the rules given in Algorithm 1:

- The first item of type 'A' is scheduled to bin $k = 1$.
- The second item of type 'A' can either enter bin $k = 1$ (Assignment $\mathcal{A}_1$) or open a new bin $k = 2$ (Assignment $\mathcal{A}_2$). The decision between these two possibilities is done by involving the next $q \geq \beta + 2$ items (i.e., at least two items of type 'C') in a best-fit fashion.

    - Extending $\mathcal{A}_1$ leads to a situation, where all items of type 'B' enter a separate bin. Then, all remaing $q - \beta$ items of type 'C' are scheduled to a new bin (with index $k = \beta + 2$).
    - Extending $\mathcal{A}_2$ leads to a situation, where all items of type 'B' enter a separate bin as well. But then, all remaing $q - \beta$ items of type 'C' are scheduled to bin number $k = 1$ (given the tie-break rule in CLH).

  Obviously, the second variant gives the lower objective value, and hence the two items of type 'A' will be placed in different bins ($k = 1$ and $k = 2$).
- The next β items (all type 'B' items) will always be assigned to a separate bin, since they cannot be packed in any of the existing bins. Moreover, any item of type 'C' will enter bin $k = 1$ according to the tie-break rule.

Altogether this produces $\beta + 2$ bins each of which requiring precisely one fire-up. In fact, the optimal solution is found in this case, and the second part of the claim is proved.

□

Obviously, both observations together ensure that for any possible q the heuristic solution obtained by CLH can be arbitrarily bad.

Theorem 9 *For any look-ahead parameter $q \in \mathbb{N}$ we have*

$$\sigma\left(CLH_q\right) = \sup_E \frac{CLH_q(E)}{OPT(E)} = \infty.$$

In particular, this result is independent of the choice of $\gamma > 0$.

Proof For any fixed $\gamma > 0$, we can use the instances $E(\alpha, \beta)$ with $\beta = q$ and consider the case $\alpha \to \infty$. □

Altogether, we see that the approximation guarantee of CLH is unbounded for every choice of q, too. In other terms, for any look-ahead parameter q an arbitrarily bad approximation guarantee can be obtained rather easily. This is a quite remarkable result since CLH was able to use information of future items to keep the objective value low, e.g., by already opening an additional bin for the item to be scheduled right now.

4 An outlook: constructive heuristics from the field of logistics

As already indicated, a wide variety of general and also temporal extensions of the classical bin packing problem has been studied in the literature, implying that the ideas of potentially promising heuristics go far beyond the algorithmic concepts investigated so far. Since one of the most important areas of bin packing applications is in the field of logistics, in this outlook we would like to exemplarily discuss some constructive methods from Baldi et al. (2019) and Crainic et al. (2021) and interpret their main ideas appropriately in terms of the TBPP-FU. The latter is mandatory, since the problems considered therein, i.e., the *generalized bin packing problem with bin-dependent item profits (GBPPI)* and the *multi-period bin packing problem (MPBPP)*, while exhibiting a few of the modeling aspects relevant in temporal bin packing, in fact differ rather strongly from our intended scenario. Before going into the details, let us briefly introduce one possible compact formulation for the TBPP-FU, because this will help to better understand the relations to the other BPP variants considered in this section. For the sake of exposition, here we just focus on the original version of model M1, introduced in Aydin et al. (2020), but we also highlight that this formulation has seen numerous improvements over the last couple of years, see Martinovic et al. (2021, 2022). In addition to the ingredients already used for the TBPP model in Sect. 2, we require the following two types of (temporal) variables:

- To display the status of server $k \in K$ at time $t \in T$ the decision variables $y_{tk} \in \{0, 1\}$ will be used in the sense that $y_{tk} = 1$ represents a positive load on server k at time t.
- The variables $w_{tk} \in \{0, 1\}$ with $k \in K$ and $t \in T_S := \bigcup_{i \in I} \{s_i\}$ contain information about the fact whether server k has been switched on at time t or not.

Then, according to Aydin et al. (2020), we obtain the

Assignment Model 1 (M1) for the TBPP-FU

$$z = \gamma \cdot \sum_{k \in K} \sum_{t \in T_S} w_{tk} + \sum_{k \in K} z_k \rightarrow \min$$

$$\text{s.t.} \quad y_{tk} \leq \sum_{i \in I_t} c_i \cdot x_{ik} \leq y_{tk} \cdot C, \qquad k \in K, t \in T, \tag{5}$$

$$\sum_{k \in K} x_{ik} = 1, \qquad i \in I, \tag{6}$$

$$x_{ik} \leq y_{s_i, k}, \qquad i \in I, k \in K, \tag{7}$$

$$y_{tk} \leq z_k, \qquad k \in K, t \in T, \tag{8}$$

$$y_{tk} - y_{t-1,k} \leq w_{tk}, \qquad k \in K, t \in T_S, \tag{9}$$

$$x_{ik} \in \{0, 1\}, \qquad i \in I, k \in K, \tag{10}$$

$$y_{tk} \in \{0, 1\}, \qquad k \in K, t \in T, \tag{11}$$

$$w_{tk} \in \{0, 1\}, \qquad k \in K, t \in T_S, \tag{12}$$

$$z_k \in \{0, 1\}, \qquad k \in K. \tag{13}$$

The objective function minimizes a weighted sum of the number of all fire-ups (first term) and all used servers (second term). Conditions (5) make sure that the capacity of any server is respected at any instant of time. Moreover, at least one job must be allocated to an active server. Constraints (6) again guarantee that each job will be executed on exactly one server. Conditions (7–9) are used to couple the different types of variables. In particular, a fire-up on server $k \in K$ must be registered at time $t \in T_S$ if this server was inactive at the previous instant of time (for simplicity referred to by $y_{t-1,k} = 0$) and has been switched on at time t (i.e., $y_{tk} = 1$).

Now, let us move forward to a more detailed discussion of the BPP variants contained in Baldi et al. (2019) and Crainic et al. (2021). Although the TBPP (and the TBPP-FU) can be coarsely covered by the label 1/U/U/S/N[ITW] in the classification[3]scheme applied in

[3] The different labels of the term 1/U/U/S/N[ITW] state that: the physical dimension of any item is equal to 1, the fixed costs of using a bin are uniform (U), the bin size is uniform (U), just a single (S) "category" of items is considered (all items are mandatory and they have to be placed), a multi-period (N) time horizon is given, and time windows are attached to the items (indicated by the additional attribute [ITW]).

Crainic et al. (2021), these two problems are different to those from Baldi et al. (2019) and Crainic et al. (2021) due to the following main reasons:

- Both GBPPI and MPBPP rely significantly on the assumption of heterogeneous bins with different capacities, but also different item-to-bin assignment costs. The latter, however, differ significantly from considering fire-ups in our cost function, since these fire-up costs cannot be determined by the individual item-to-bin assignment costs, but depend on the overall pattern assigned to the respective bin.
- In both, the GBPPI and the MPBPP, not all items have to be assigned to the given bins. In the case of the GBPPI, there are so-called *non-compulsory items*, while in the case of the MPBPP, there is the possibility to use so-called *spot-market bins* for packing the given items. Both possibilities do not comply with the specifications of temporal bin packing.
- In GBPPI there is no time dimension at all. Consequently, it is not a generalization of TBPP.
- In MPBPP, a time dimension exists at least implicitly. This means that the assignment of an item (to a bin) must happen within a certain time interval, but the items themselves do not possess a lifespan in our sense (or formulated differently: reside only exactly one time step in the bin). Furthermore, the bins in the MPBPP do not represent two-dimensional objects (in the capacity-time plane), but may also only be used at an arbitrarily chosen point in time (then as a one-dimensional bin). Consequently, the TBPP (and also the TBPP-FU) are not special cases of the MPBPP.

All these reasons contribute to the fact that we cannot directly apply the heuristics presented in Baldi et al. (2019) and Crainic et al. (2021), because many of the sortings used therein have no meaning in our setting, and the decisions to be made would have to be interpreted appropriately. Therefore, in transferring the heuristics from Baldi et al. (2019) and Crainic et al. (2021), we restrict ourselves to essential core ideas that did not appear in our previous approaches yet, but (as a result of the previous argumentation) we use heuristics with sometimes significant modifications compared to the algorithms in Baldi et al. (2019) and Crainic et al. (2021). Altogether, as a result of this literature study the following general concepts have been identified as potentially promising:

- *Lowest Cost (LC)* with limited lookahead (in light of the heuristics called *Best Profitable* and *Best Assignment* in Subsect. 4.1. in Baldi et al. (2019)).
- *Bin Best (BB)* (in light of bin-centric heuristics from Subsect. 5.2 in Crainic et al. (2021)).
- *Best Clique (BC)* (in light of decomposition-based heuristics from Sect. 6 in Crainic et al. (2021)).

Let us start with *Lowest Cost (LC)* from Baldi et al. (2019), which would work as illustrated in Algorithm 2 if applied to our scenario[4]:

[4] In fact, the heuristics called *Best Profitable* and *Best Assignment* in Baldi et al. (2019) make use of some problem-specific features (like different bin types, fixed item-to-bin assignment costs, non-compulsory items, etc.) which are not relevant to the TBPP-FU introduced here. As a consequence of that, some steps of these algorithms (like checking for cost-reducing packing pattern swaps between two used bins in the post-processing) are not meaningful in our case, while other steps have to be interpreted "in the sense of the TBPP-FU cost function". In particular, there is no difference between BP and BA for the TBPP-FU.

Algorithm 2 Lowest Cost (LC) for the TBPP-FU

```
Input: Item list ordered by non-decreasing starting times $s_i$, look-ahead parameter $N \in \mathbb{N}$ with $N \ll n$.
1: Place the first item in the first bin $k = 1$. Remove this item from $I$.
2: while $I \neq \emptyset$ do
3:     Label the first $N$ (or less, if $|I| < N$) items from $I$ by $r = 1, \ldots, \min\{N, |I|\}$.
4:     Execute the following step independently for any such item: Compute the bin $B^\star(r)$ which is able to
       accommodate item $r$ while inducing the lowest costs (breaking ties by the lowest bin index). If there
       is no such bin, pack item $r$ to a new bin and call this bin $B^\star(r)$.
5:     Choose the item $r^\star$ that generated the lowest costs (breaking ties by the lowest item index) in the
       previous step and place it to the corresponding bin $B^\star(r^\star)$. Remove item $r^\star$ from $I$.
6: end while
Output: heuristic solution with objective value $LC_N(E)$.
```

In this algorithm, the item list is iteratively reduced by choosing an item which would currently(!) add the lowest costs (including both, new fire-ups and new servers) to the objective function. To be more precise, this item has to be chosen from a sublist involving the next $N \ll n$ items, meaning that a typically rather limited look-ahead (see Baldi et al. (2019)) is allowed. Note that, in our scenario, we have to add tie-break rules, since there are only three possible cost terms that can appear when adding one item to a bin. Indeed, an item can locally generate no costs at all (neither causing a fire-up nor a new bin), γ cost units (causing a fire-up in an existing bin), or $(1 + \gamma)$ cost units (opening a new bin).

Remark 5 Although there are at least two similarities between CLH and LC, namely a look-ahead strategy and the best-cost argumentation, both heuristics are fundamentally different. In particular, CLH always places the currently lowest-indexed item i. The decision where to place this item is done by (at least implicitly) assigning the items from the look-ahead to the existing servers. Contrary to that, LC just places some item $r^\star$ from the currently considered sublist (of length N). Moreover, the remaining items from the look-ahead are not involved (in the sense of being scheduled together with $r^\star$) when evaluating the costs of placing an item.

The next result shows that the approximation guarantee of Algorithm 2 can be arbitrarily bad.

Theorem 10 *For any look-ahead parameter $N \geq 1$ we have*

$$\sigma\,(LC_N) = \sup_E \frac{LC_N(E)}{OPT(E)} = \infty.$$

In particular, this result is independent of the choice of $\gamma > 0$.

Proof By way of example, let us consider an instance $E(\alpha, \beta)$ with $\beta \in \mathbb{Z}_+$ and sufficiently large α, so that $N \ll n$ holds. Obviously, the first item of type 'A' is placed in bin $k = 1$ as an initialization. When making the decision, where to place the second item of type 'A', we have to consider the next N items individually. This sublist of items contains one item of type 'A', possibly together with some items of type 'B' and 'C' (depending on the precise choice of β and N). Now, we have the following item-specific cost-minimal assignments:

- The second item of type 'A' has to go in bin $k = 1$, where it adds zero units to the cost function.
- Any item of type 'B' has to open a new bin, producing $1 + \gamma$ cost units.
- Any item of type 'C' is scheduled to $k = 1$ without generating further costs.

According to the tie-break rules, the second item of type 'A' is added to bin $k = 1$. Given the characteristics of the remaining items, Algorithm 2 will schedule them in exactly the same way as CLH_q before. So, we end up with an objective value of $\beta + 2 + \gamma(\beta + 1 + \alpha)$. For $\alpha \to \infty$ the claim follows. □

While the heuristics studied so far have placed exactly one item per step and, thus, could not predict the interaction with items yet to be placed in the future, we would now like to consider two more approaches originating from concepts proposed in Crainic et al. (2021). These approaches are *offline algorithms*, since they require knowledge of all input data (from the beginning) and assign a subset of items in one or every iteration.

Let us start with a bin-oriented heuristics called Best Bin (BB), see Algorithm 3:

Algorithm 3 Best Bin (BB) for the TBPP-FU

```
Input: Instance $E$ of the TBPP-FU.
1: while $I \neq \emptyset$ do
2:     According to a decision criterion, find the best items-to-bin assignment (using the items of $I$).
3:     Remove the items required and update set $I$.
4: end while
Output: heuristic solution with objective value $BB(E)$.
```

In fact, the above heuristic iteratively constructs promising packing patterns and fills exactly one bin per iteration, meaning that a temporal knapsack problem with fire-ups is considered in any step. Of course, there is wide variety of potential decision criteria which could be used in the main part of Algorithm 3. According to the objective function of the TBPP-FU, the most important objective surely is that any of the obtained items-to-bin assignments should possess a (preferably) low number of fire-ups. However, this goal alone is not sufficient to obtain good-quality solutions, in general. This is because, in any step, assigning just one item to the currently considered bin would be optimal. Thus, it is important to not only keep the number of fire-ups low, but to try to assign further items to the same bin while not (substantially) changing its contribution to the overall objective function. Among others, the latter can be achieved by adding (i) the total area of items assigned or (ii) the total number of items assigned to the objective function and formulating the decision criterion (in Algorithm 3) in a weighted-sum fashion. Given two weighting parameters $K_1, K_2 \in (0, 1)$ with $K_1 + K_2 = 1$, a more reasonable score to rank[5] various items-to-bin assignments can be defined as

$$K_1 \cdot \sum_{t \in T} w_t - K_2 \cdot \sum_{i \in I} c_i \cdot (e_i - s_i) \cdot x_i \qquad \text{or} \qquad K_1 \cdot \sum_{t \in T} w_t - K_2 \cdot \sum_{i \in I} x_i,$$

with $x_i \in \{0, 1\}$ denoting whether item $i \in I$ is part of the allocation or not, and $w_t \in \{0, 1\}$ stating whether a fire-up is perceived at starting time $t \in T_S$. By that, the costs of fire-ups are appropriately included in our heuristic. Moreover, using such a decision criterion, the probability of obtaining single-item bins (by Algorithm 3) is reduced considerably.

The following result shows that also the bin-oriented heuristic can be arbitrarily bad.

Theorem 11 *For any valid choice of the parameters K_1 and K_2, the worst-case performance ratio of Algorithm 3 is unbounded, i.e.,*

$$\sigma(BB) = \sup_E \frac{BB(E)}{OPT(E)} = \infty.$$

[5] Here, a lower score would mean that the packing pattern is more promising.

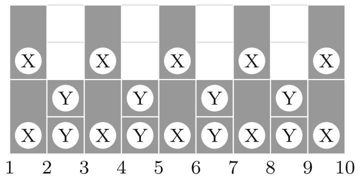

Fig. 3 An illustration of $E(p)$ for $p = 5$

In particular, this result is independent of the choice of $\gamma > 0$.

Proof Let us consider a family of instances $E(p)$ characterized by $p \geq 2$ and defined in the following way:

- We use a bin capacity of $C = 2$.
- We have $2p$ items (called type 'X'), consisting of p groups of two items each. The items of group $i \in \{1, \dots, p\}$ satisfy $c_i = 2$ and $[s_i, e_i) = [2i - 1, 2i)$.
- We have $2(p - 1)$ items (called type 'Y'), consisting of $p - 1$ groups of two items each. The items of group $i \in \{1, \dots, p - 1\}$ satisfy $c_i = 1$ and $[s_i, e_i) = [2i, 2i + 1)$.

For clarity, an example with $p = 5$ is illustrated in Fig. 3.

We now study the optimal and heuristic solution belonging to $E(p)$:

- Obviously, an optimal solution is given by using two servers with one X-item of each of the p groups. Moreover, on any of the two servers the temporal gap between two successive X-items can be bridged by precisely one Y-item. This leads to two bin that are executed without interruption, i.e., we have
$$OPT(E(p)) = 2 \cdot (1 + \gamma).$$
- When using Algorithm 3, the first bin would be filled completely (identical to the lower half of Fig. 3), because this assignment just produces one fire-up, but uses the largest possible area or the largest number of items, respectively. In any possible schedule, the remaining items will produce exactly p fire-ups and use at least one additional bin (possibly more bins, depending on the precise choice of K_1 and K_2), so we end up with
$$BB(E(p)) \geq 2 + \gamma \cdot (1 + p).$$

Altogether this leads to:

$$\frac{BB(E(p))}{OPT(E(p))} \geq \frac{2 + \gamma \cdot (1 + p)}{2 \cdot (1 + \gamma)}$$

which tends to infinity for $p \to \infty$. This proves the claim. □

The last heuristic to be discussed here is based on the idea of a temporal decomposition of the considered instance. This concept of focussing on the items available at different points in time is inspired by the investigations in Sect. 6 in Crainic et al. (2021), and one possible implementation (in the sense of the TBPP-FU) can be found in Algorithm 4. Recall that T_S denotes the set of starting points of a given instance. To better understand the key ideas of the heuristic, we first repeat an important definition from the TBPP literature.

Definition 2 (Dell'Amico et al. (2020)) Let E be an instance of the TBPP and let $t_1 < t_2 \in T_S$ follow each other directly in the chronologically ordered set T of all time instants. If t_2 is not also an ending time, then t_1 *is dominated by* t_2. The set of all non-dominated starting times is referred to as $T_S^{nd} \subseteq T_S$.

Note that there is a bijection between the non-dominated starting times and the maximum cliques of the interval graph belonging to E, see Martinovic et al. (2023), implying a canonical order of the maximum cliques. Moreover, the latter can be efficiently determined in polynomial time, see Biedl (2005) or Algorithm 1 in Furini (2011), for an implementation with $\mathcal{O}(n^2)$ time.

The details of the clique-based heuristic called Best Clique (BC) can be found in Algorithm 4:

Algorithm 4 Best Clique (BC) for the TBPP-FU

```
Input: Instance $E$ of the TBPP-FU with an item list ordered by non-decreasing starting times $s_i$.
1: Compute the maximum cliques $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$ of $E$.
2: for all $C_l \in \mathcal{C}$ do
3:     Solve the bin packing problem related to $C_l$. Let $z_l^\star$ denote the number of bins required to pack the
       items of $C_l$.
4: end for
5: Let $l^\star$ indicate the index of the clique having the largest value $z_l^\star$ (breaking ties by the lowest running index
   $l$). Assign the items of $C_{l^\star}$ according to the solution obtained by solving the bin packing problem.
6: Starting at $C_{l^\star}$, iteratively move to the left-neighbor clique (until there is no left neighbor anymore) and
   assign the unpacked items (of that respective clique) to the existing bins in a best-cost fashion (breaking
   ties by assigning the item in a best-fit fashion among the cost-minimal candidates, and, if a further criterion
   is required, choosing the lowest-indexed bin). If an item does not fit into the existing bins, a new bin has
   to be opened to accommodate this item.
7: Starting at $C_{l^\star}$, iteratively move to right-neighbor clique (until there is no right neighbor anymore) and
   assign the unpacked items (of that respective clique) to the existing bins in a best-cost fashion (breaking
   ties by assigning the item in a best-fit fashion among the cost-minimal candidates and, if a further criterion
   is required, choosing the lowest-indexed bin). If an item does not fit into the existing bins, a new bin has
   to be opened to accommodate this item.
Output: heuristic solution with objective value $BC(E)$.
```

The idea of this algorithm is to first find the most restrictive point in time (i.e., the maximum clique is likely to be responsible for opening the most bins) and to assign the items of that clique in a locally optimal way. Then, in both directions, the new items of the neighboring cliques are iteratively added to that fundament in a best-cost fashion trying to keep the number of required bins and the number of fire-ups low. Again, the local(!) costs of adding an item to some existing schedule can be 0 (item does neither open a new bin nor leads to a new fire-up on the server), γ (item leads to an additional fire-up on the server), or $1 + \gamma$ (item is placed on an empty server). As a consequence of that, a two-staged tie-break system was added to Algorithm 4.

Remark 6 Given its focus on maximum cliques, the BC heuristic strongly exploits the temporal structure of a given instance (e.g., by focussing on the non-dominated starting times). Moreover, adding an item to a bin is done in a best-cost fashion, so that both aspects of the objective function (number of bins and number of fire-ups) are always present in the decision-making process.

The following result shows that also the clique-based heuristic can be arbitrarily bad.

Theorem 12 *We have*

$$\sigma(BC) = \sup_E \frac{BC(E)}{OPT(E)} = \infty.$$

In particular, this result is independent of the choice of $\gamma > 0$.

Proof Let us consider an instance $\widetilde{E}(\alpha, 0)$ with $C = 2$ that is similar to $E(\alpha, 0)$ with $\alpha \geq 2$, but where the first item of type 'C' (labelled $i = 1$ in Fig. 1) is replaced by two items (referred to as type 'D') with $c_i = 2$ and $[s_i, e_i) = [1, 2)$.

- Since $\alpha \geq 2$ holds, an optimal solution requires four bins. Two of these bins contain precisely one item of type 'D'. The third bin is filled with one item of type 'A', and the fourth bin contains the remaining items (one item of type 'A' and all items of type 'C'). In total, this leads to four bins with one fire-up each, i.e., we have
$$OPT(\widetilde{E}(\alpha, 0)) = 4 \cdot (1 + \gamma).$$
- Obviously, any odd t represents a non-dominated starting time (maximum clique). The clique for $t = 1$ contains two items of types A and D each, the cliques for $t \neq 1$ contain one item of type 'C' together with two items of type 'A'. Hence, C_1 leads to the largest objective value for the classic bin packing problem ($z_1^\star = 3$), and our starting configuration consists of one bin ($k = 1$) with both items of type 'A' (so this bin cannot accept any further item), and two bins ($k = 2, 3$) with one item of type 'D' each. Since C_1 was the left-most clique, we can just proceed in right direction. In every iteration (that is, in any clique considered), precisely one item of type 'C' is unassigned, and so it has to enter the given bins in a best-cost fashion. According to the tie-break rules, all these items are added to bin $k = 2$.
In the end, we have two bins with exactly one fire-up ($k = 1$ and $k = 3$) and one bin with a total of α fire-ups ($k = 2$). This leads to
$$BC(\widetilde{E}(\alpha, 0)) = 3 + \gamma \cdot (2 + \alpha).$$

Hence, for $\alpha \to \infty$ the worst-case performance ratio becomes arbitrarily large, and the claim is proved. □

Altogether, we have shown that neither the ideas obtained from constructive heuristics in the field of logistics lead to approaches with bounded approximation guarantee. This observation is remarkable, since we have now dealt with a wide variety of online, semi-online, and offline heuristics – and none of them was theoretically able to handle fire-ups efficiently.

5 Conclusions

In this article, we considered heuristic approaches from the literature for two neighboring optimization problems, the TBPP and the TBPP-FU. For the former, we briefly repeated some important results obtained from publications on dynamic bin packing, because according to our impression these contributions have not yet made their way into the most recent scientific discussion on the TBPP. For the TBPP-FU, in a first step we were able to show that the worst-case performance ratio of the former TBPP heuristics is unbounded when fire-ups have to be considered. Having identified the critical features of these algorithms, we observed that they are not present in the only heuristic specifically designed for the TBPP-FU in the literature, that is, CLH from Aydin et al. (2020). However, despite the promising modifications contained in CLH, the approximation guarantee of this heuristic is unbounded, too. As an outlook, similar results were shown for further constructive heuristics the ideas of which originating from a logistics application, see Baldi et al. (2019) and Crainic et al. (2021). Remarkably, all results are independent of the scaling parameter γ and the look-ahead parameter q (resp. N). Altogether, this work represents the first systematic framework to study theoretical properties of heuristic approaches for the TBPP-FU, showing some previously unknown differences between the two temporal problems under consideration. On the other hand, our contributions inevitably lead to a new open challenge for future research in cutting and packing, that is: How to construct a (preferably simple) heuristic for the TBPP-FU, the performance ratio of which can be shown to be bounded?

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors declare that they do not have any conflicts of interest.

References

Aydin, N., Muter, I., & Ilker Birbil, S. (2020). Multi-objective temporal bin packing problem: An application in cloud computing. *Computers & Operations Research, 121*, 104959.

Baldi, M. M., Manerba, D., Perboli, G., & Tadei, R. (2019). Generalized bin packing problem for parcel delivery in last-mile logistics. *European Journal of Operational Research, 274*(3), 990–999.

Bartlett, M., Frisch, A. M., Hamadi, Y., Miguel, I., Tarim, S., & Unsworth, C. (2005). The temporal knapsack problem and its solution. *Lecture Notes in Computer Science, 3524*, 34–48.

Biedl, T. (2005). *Graph-theoretic algorithms*. Lecture notes: University of Waterloo.

Caprara, A., Furini, F., & Malaguti, E. (2013). Uncommon Dantzig-Wolfe reformulation for the temporal knapsack problem. *INFORMS Journal on Computing, 25*(3), 560–571.

Chan, J.W.-T., Lam, T.-W., & Wong, P. W. H. (2008). Dynamic bin packing of unit fraction items. *Theoretical Computer Science, 409*, 521–529.

Chan, J.W.-T., Wong, P. W. H., & Yung, F. C. C. (2009). On dynamic bin packing: an improved lower bound and resource augmentation analysis. *Algorithmica, 53*, 172–206.

Coffman, E. G., Jr., Garey, M. R., & Johnson, D. S. (1983). Dynamic bin packing. *SIAM Journal on Computing, 12*(2), 227–258.

Coffman, E. G., Jr., Garey, M. R., & Johnson, D. S. (1987). Bin packing with divisible item sizes. *Journal of Complexity, 3*(4), 406–428.

Crainic, T. G., Fomeni, F. D., & Rei, W. (2021). Multi-period bin packing model and effective constructive heuristics for corridor-based logistics capacity planning. *Computers & Operations Research, 132*, 105308.

de Cauwer, M., Mehta, D., & O'Sullivan, B. (2016). The temporal bin packing problem: An application to workload management in data centres. Proceedings of the 28th IEEE International Conference on Tools with Artificial Intelligence, pp. 157–164

Dell'Amico, M., Furini, F., & Iori, M. (2020). A branch-and-price algorithm for the temporal bin packing problem. *Computers & Operations Research, 114*, 104825.

Delorme, M., Iori, M., & Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research, 255*, 1–20.

Dósa, G.,& Sgall, J. (2013). First Fit bin packing: A tight analysis. 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013), pp. 538–549

Dósa, G., & Sgall, J. (2014). Optimal analysis of best fit bin packing. *Lecture Notes in Computer Science, 8572*, 429–441.

Fettweis, G., Dörpinghaus, M., Castrillon, J., Kumar, A., Baier, C., Bock, K., Ellinger, F., Fery, A., Fitzek, F., Härtig, H., Jamshidi, K., Kissinger, T., Lehner, W., Mertig, M., Nagel, W., Nguyen, G. T., Plettemeier, D., Schröter, M., & Strufe, T. (2019). Architecture and advanced electronics pathways towards highly adaptive energy-efficient computing. *Proceedings of the IEEE, 107*(1), 204–231.

Flammini, M., Monaco, G., Moscardelli, L., Shachnai, H., Shalom, M., Tamir, T., & Zaks, S. (2010). Minimizing total busy time in parallel scheduling with application to optical networks. *Theoretical Computer Science, 411*, 3553–3562.

Frangioni, A., & Gentile, C. (2006). Solving nonlinear single-unit commitment problems with ramping constraints. *Operations Research, 54*(4), 767–775.

Furini, F. (2011). Decomposition and reformulation of integer linear programming problems. PhD thesis, Università di Bologna.

Gschwind, T., & Irnich, S. (2017). Stabilized column generation for the temporal knapsack problem using dual-optimal inequalities. *OR Spectrum, 39*, 541–556.

Johnson, D. S. (1973). Near-optimal bin packing algorithms. PhD dissertation, Massachusetts Institute of Technology

Marcotte, O. (1983). Topics in combinatorial packing and covering. Technical Report No. 568, Cornell University

Martinovic, J. (2022). A note on the integrality gap of cutting and skiving stock instances. *Why 4/3 is an upper bound for the divisible case? 4OR*, 20, 85–104.

Martinovic, J., Strasdat, N. (2022). Theoretical Insights and a New Class of Valid Inequalities for the Temporal Bin Packing Problem with Fire-Ups. Preprint MATH-NM-01-2022, Technische Universität Dresden (http://www.optimization-online.org/DB_HTML/2022/02/8791.html)

Martinovic, J., Strasdat, N., & Selch, M. (2021). Compact integer linear programming formulations for the temporal bin packing problem with fire-ups. *Computers & Operations Research, 132*, 105288.

Martinovic, J., Strasdat, N., Valério de Carvalho, J. M., & Furini, F. (2022). Variable and constraint reduction techniques for the temporal bin packing problem with fire-ups. *Optimization Letters, 16*, 2333–2358.

Martinovic, J., Strasdat, N., Valério de Carvalho, J. M., & Furini, F. (2023). A combinatorial flow-based formulation for temporal bin packing problems. *European Journal of Operational Research, 307*(2), 554–74.

Scheithauer, G. (2018). Introduction to cutting and packing optimization–problems, modeling approaches, solution methods. International Series in Operations Research & Management Science 263, Springer, 1.Edition