Optimization Letters (2022) 16:2333–2358
https://doi.org/10.1007/s11590-021-01825-x

ORIGINAL PAPER

Variable and constraint reduction techniques for the temporal bin packing problem with fire-ups

John Martinovic[1] · Nico Strasdat[1] · José Valério de Carvalho[2] · Fabio Furini[3]

Received: 20 May 2021 / Accepted: 3 November 2021 / Published online: 4 December 2021

Abstract
The aim of this letter is to design and computationally test several improvements for the compact integer linear programming (ILP) formulations of the temporal bin packing problem with fire-ups (TBPP-FU). This problem is a challenging generalization of the classical bin packing problem in which the items, interpreted as jobs of given weight, are active only during an associated time window. The TBPP-FU objective function asks for the minimization of the weighted sum of the number of bins, viewed as servers of given capacity, to execute all the jobs and the total number of fire-ups. The fire-ups count the number of times the servers are activated due to the presence of assigned active jobs. Our contributions are effective procedures to reduce the number of variables and constraints of the ILP formulations proposed in the literature as well as the introduction of new valid inequalities. By extensive computational tests we show that substantial improvements can be achieved and several instances from the literature can be solved to proven optimality for the first time.

Keywords Cutting and packing · Temporal bin packing problem · Fire-ups · Integer linear programming

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

Nico Strasdat
nico.strasdat@tu-dresden.de

José Valério de Carvalho
vc@dps.uminho.pt

Fabio Furini
fabio.furini@uniroma1.it

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

[2] Departamento de Produção e Sistemas, Universidade do Minho, Braga, Portugal

[3] Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Rome, Italy

1 Introduction

The *temporal bin packing problem* (TBPP) introduces a temporal dimension to the classical *bin packing problem* (BPP), see [11,18,20], by associating to the items time windows in which they are active. Formally, given a set of *items* (or *jobs*), the TBPP asks for determining the minimum number of *bins* (or *servers*) of given capacity to execute all jobs. Each job is characterized by a *size* (or *resource demand*) and a *lifespan* (time window in which the job is active), a jobs-to-servers assignment is feasible if and only if the capacity of the servers is respected at any instant of time. The TBPP is a challenging problem with a high practical and theoretical interest which has been recently introduced in the literature, see [9,10]. It belongs to the rich family of cutting and packing problems, object of intense research in the last decades. The TBPP can also be interpreted as a high-dimensional vector packing problem [7,19] and it partially shares the mathematical structures of two-dimensional packing problems, where the time can be seen as one of the dimensions and the other one is related to the capacity of the servers. We also refer the interested reader to [13,16] for the strip packing versions of these problems.

Another problem closely related to the TBPP is the *temporal knapsack problem* (TKP). The TKP asks for determining a maximum-profit subset of jobs (also active for a given time window) which can be executed by a single server, see [2,4]. As far as the exact approaches to solve the TKP to proven optimality are concerned, this problem is usually tackled by a Dantzig-Wolfe reformulation and branch-and-price algorithms [5,6] or by dynamic programming algorithms [8].

The TBPP is introduced in an application-oriented article, see [9], dealing with efficient workload server management in data centers–one of the key challenges in light of the ever-growing energy demand of the IT industry [1,12,14]. As far as the TBPP solution methods are concerned, we refer the reader to [10], an article which presents several heuristic and exact approaches. The state-of-the-art exact algorithm for the TBPP is a branch-and-price algorithm, which exploits in preprocessing a plethora of heuristic algorithms (see [10] for further details).

In the TBPP, the quality of a jobs-to-servers assignment is evaluated solely by the number of servers in use. However, several recent publications point out that the specific operating mode of the servers is also crucial. This means, in particular, that an inactive server can be temporarily put into a sleep mode (or can be completely shut down) for the purpose of saving energy. In this case the server must then be turned on again, if required. A server restart is called a *fire-up*, see also Fig. 1 for a graphical representation, and–from the perspective of energy-efficiency–this naturally leads to a second optimization goal, introduced in [3]. The authors of that article propose two compact ILP formulations (denoted M1 and M2) taking into consideration two different objectives: (i) the number of servers and (ii) the number of fire-ups. This new objective function is a weighted-sum, using the input parameter $\gamma > 0$ to scale the contribution of the fire-ups. In this way, a multi-objective variant of the TBPP has been proposed in [3], hereinafter referred to as the *temporal bin packing problem with fire-ups* (TBPP-FU). The models M1 and M2 of [3] are based on the classic Kantorovich-type structure for the BPP, see [15], and the experiments show that the proposed exact solution method using these models is very challenging from a computational

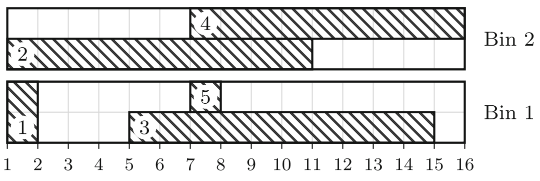

Fig. 1 An exemplary assignment of five jobs to two servers. (This assignment is optimal for the instance appearing in Example 1 with $\gamma = 1$)

point of view. In order to quickly find good-quality heuristic solutions, a *constructive look-ahead heuristic* (CLH), together with an advanced recovery algorithm based on mathematical programming techniques, is also presented in [3].

In that article, an interesting insight of the multi-objective function is also shown: for rather small values of γ, i.e., for $\gamma \leq 1/n$ (where n is the number of jobs), it is possible to considerably reduce the size of the ILP formulations by exploiting the information given by the heuristic solutions. However, the important question on how to reduce the models in the general case of arbitrary values of γ is still an open problem which we tackle in this letter.

Very recently, in [17] several methods for improving the TBPP-FU ILP models have been proposed. These techniques generate substantial performance gains, including also the possibility of efficiently handling larger values of γ. In addition, a third formulation (called M3) is proposed which is not based on the classical job-to-server assignment variables. More in details, M3 is based on a job-to-job relation and it allows to a priori discard some infeasible solutions by removing a set of variables. As a result, model M3 is a more compact ILP formulation (compared to M1 and M2) which is particularly effective for fast calculations. However, from an overall point of view, the tests in [17] show that the "optimized" version of M1 remains on average the state-of-the-art approach in terms of instances solved to proven optimality.

In this article, we aim to enrich and complete the structural analysis of the compact ILP formulations for the TBPP-FU. To this end, we present new methods to improve the ILP models which lead to better numerical results. More in details, we show how to transfer parts of the previously mentioned favorable properties of M3, i.e., mainly a small model size and the possibility to easily account for job incompatibilities, to M1 and M2, thus obtaining very robust and more powerful compact formulations. The main contributions of our investigation are:

- We optimize and reduce the set of constraints of M1 by removing redundancies and tightening some conditions.
- We present a new class of valid inequalities for M1 and M2 mimicking the inherent property of M3 to avoid forbidden item pairs.
- We theoretically show how heuristic-based information can be used for arbitrary choices of γ, generalizing the results of the literature. A side effect of investigating the heuristic used for that purpose is that we can also pass a good-quality starting point to the ILP solver.

The remainder of this letter is structured as follows: In the next section we introduce the notation and the M1 and M2 formulations from the literature. In the core Sect. 3, we present the new reduction procedures and the new family of valid inequalities.

Finally, extensive computational tests are reported in Sect. 4, aiming at computationally evaluating the beneficial effect of the newly proposed techniques.

2 Preliminaries and basic models

In this section, we present the most important terminologies and notations, as well as a brief overview of the current ILP formulations from the literature. Let us consider $n \in \mathbb{N}$ *items* (*jobs*), each of which possessing a *resource demand* (*item size*) $c_i \in \mathbb{N}$ that is active only in the interval $[s_i, e_i)$ formed by the *starting time* $s_i \in \mathbb{N}$ and the *ending time* $e_i \in \mathbb{N}, i \in I := \{1, \dots, n\}$. Then, these items have to be assigned to *bins* (*servers*) of capacity $C \in \mathbb{N}$ so that a weighted sum consisting of (i) the number of servers required and (ii) the number of fire-ups is minimized, where the second term is scaled by a weighting parameter $\gamma > 0$. Let $K := \{1, \dots, n\}$ denote the set of servers and let $T := \bigcup_{i \in I} \{s_i, e_i\}$ and $T_S := \bigcup_{i \in I} \{s_i\}$ collect the relevant instants of (starting) times. Moreover, we assume the items to be ordered with respect to non-decreasing starting times (where ties are broken arbitrarily), and we use $I_t := \{i \in I \mid t \in [s_i, e_i)\}$ to indicate the active jobs at time $t \in T$.

Example 1 Let us consider an instance with servers of capacity $C = 2$ and $n = 5$ items having resource demands $c = (2, 1, 1, 1, 1)$, starting times $s = (1, 1, 5, 7, 7)$, and ending times $e = (2, 11, 15, 16, 8)$. Then, for $\gamma = 1$, an optimal solution requires two servers and three fire-ups as depicted in Fig. 1.

In total, three different compact models for the TBPP-FU have so far been formulated in the literature. To reiterate, the original publication [3] contained two ILP formulations (called M1 and M2) which were based on classic job-to-server assignment variables. Some first reduction methods together with a new third exact approach (called M3) have been presented in [17], all of which contributed to significantly better numerical results. To facilitate understanding the ILP formulations, here we only present the raw versions from [3] in some detail. However, we emphasize that all of our contributions presented later are directly applied to the models from [17], which correspond to the current state of the literature. In contrast, a formal introduction of M3 is skipped since our new reductions either specifically address the set of servers to be modeled (which, however, is not explicitly contained in M3) or exclude combinations of items (which is automatically done in M3 through job-to-job coupling).

Remark 1 Nevertheless, we will also consider an improved version of M3 in the numerical part, but its differences with the version from [17] are very general and therefore traceable even without explicit knowledge of the model itself.

To state M1, let us consider the following four types of binary variables:

- We have $z_k = 1$ if and only if server $k \in K$ is used.
- We set $x_{ik} = 1$ if and only if job $i \in I$ runs on server $k \in K$.
- We use $y_{tk} = 1$ if and only if server $k \in K$ is active at time $t \in T$.
- We have $w_{tk} = 1$ if and only if server $k \in K$ is activated at time $t \in T_S$.

Consequently, the z-variables measure the number of servers required whereas the w-variables are responsible for counting the fire-ups. The original version of M1, as presented in [3], then reads as follows

Model 1 (M1)

$$\min \sum_{k \in K} \left(z_k + \gamma \sum_{t \in T_S} w_{tk} \right)$$

$$\text{s.t.} \quad y_{tk} \leq \sum_{i \in I_t} c_i x_{ik} \leq y_{tk} C, \qquad k \in K, t \in T, \tag{1}$$

$$\sum_{k \in K} x_{ik} = 1, \qquad i \in I, \tag{2}$$

$$x_{ik} \leq y_{s_i,k}, \qquad i \in I, k \in K, \tag{3}$$

$$y_{tk} \leq z_k, \qquad k \in K, t \in T, \tag{4}$$

$$y_{tk} - y_{t-1,k} \leq w_{tk}, \qquad k \in K, t \in T_S, \tag{5}$$

$$x_{ik} \in \{0,1\}, \qquad i \in I, k \in K, \tag{6}$$

$$y_{tk} \in \{0,1\}, \qquad k \in K, t \in T, \tag{7}$$

$$w_{tk} \in \{0,1\}, \qquad k \in K, t \in T_S, \tag{8}$$

$$z_k \in \{0,1\}, \qquad k \in K. \tag{9}$$

The objective function minimizes a weighted sum of the aforementioned criteria. Constraints (1) make sure that the capacity of an active server is respected (right hand side), and that an empty server is deactivated (left hand side). Conditions (2) demand that any job is assigned exactly once, while linking the different types of variables is done by Restrictions (3)–(5). In particular, (5) is responsible for recognizing a fire-up in precisely those cases where the considered server is currently active, but was inactive at the preceding instant of time (indicated by $t-1$ in a symbolic way).

The second formulation M2 appearing in [3] addresses the temporal aspect of the problem in a less explicit way, e.g., by not making use of the y-variables measuring the activity of the servers. Instead, however, the sets $\delta_i := \{ j < i \mid s_i < e_j \}$ and $\delta_i^+ := \{ j < i \mid s_i \leq e_j \}$, $i \in I$, gathering jobs being active at s_i (see δ_i and δ_i^+) or just having finished at s_i (only δ_i^+) are required. With these ingredients, M2 from [3] is given by:

Model 2 (M2)

$$\min \sum_{k \in K} \left(z_k + \gamma \sum_{t \in T_S} w_{tk} \right)$$

$$\text{s.t.} \quad \sum_{j \in \delta_i} c_j x_{jk} + c_i x_{ik} \leq C z_k, \qquad i \in I, k \in K \tag{10}$$

$$\sum_{k \in K} x_{ik} = 1, \qquad i \in I, \tag{11}$$

$$x_{ik} \leq z_k, \qquad i \in I, k \in K, \tag{12}$$

$$w_{s_i,k} \geq x_{ik} - \sum_{j \in \delta_i^+} x_{jk}, \qquad i \in I, k \in K, \tag{13}$$

$$x_{ik} \in \{0, 1\}, \qquad i \in I, k \in K, \tag{14}$$

$$w_{tk} \in \{0, 1\}, \qquad t \in T_S, k \in K, \tag{15}$$

$$z_k \in \{0, 1\}, \qquad k \in K. \tag{16}$$

The objective function and Conditions (11) already appeared in exactly the same way in M1, whereas Constraints (10) again ensure that the server capacity is respected. Restrictions (12)–(13) are responsible for linking the different types of variables. In particular, Conditions (13) state that a fire-up at time s_i has to be perceived on server k, if item i, but none of the items from δ_i^+, has been placed on it.

Besides being relatively large in size, the original versions of M1 and M2 possess some structural drawbacks:

(i) The solution space is highly symmetric.
(ii) The LP relaxation is rather poor.
(iii) The set K is much larger than required in an optimal solution.

Note that the first two problems are mainly related to the Kantorovich-type structure of the models and were already successfully addressed in parts in [17]. Without going into too much detail (for this, we refer the reader to the contributions of [17]), this was done primarily by:

(R1) Renumbering the servers so that only index pairs (i, k) from the set $\Delta := \{(i, k) \mid k \leq i\}$ have to be considered. This also made it possible to move to server-dependent time sets

$$T(k) = \bigcup_{i \geq k} \{s_i, e_i\} \quad \text{and} \quad T_S(k) = \bigcup_{i \geq k} \{s_i\},$$

which helped to also save some of the y- and w-variables.

(R2) Introducing valid inequalities $z_k \leq \sum_{t \in T_S(k)} w_{tk}$ for any $k \in K$ implying at least one fire-up on every server. In particular, this prevents the two variable types in the objective function from becoming arbitrarily small independently of each other.

(R3) Lifting the item sizes c_i, $i \in I$, to possibly tighten Conditions (1) and (10).

Furthermore, by additional minor reductions (referred to as (R4) and (R5) in [17]), some redundancies within the set of constraints could be eliminated in M2. Based on numerical tests, it was shown that the listed techniques lead to significant improvements of the compact models, with (R1) and (R2) proving to be particularly valuable for the benchmark instances considered. This was due mainly to the fact that the size of the models could be reduced by roughly 50% (both in terms of variables and constraints), but also to significantly better LP bounds. For this reason, it seems worthwhile to us to explore further reduction possibilities (for the approaches from [17]) and present them in the next section, thus arriving at the somewhat "best possible" version of the compact formulations M1 and M2 (and, in the light of Remark 1, also M3).

3 New reduction methods

The new reduction methods to be proposed in this section can be roughly divided into three categories:

(a) "optimizing" the set of constraints,
(b) adding new valid inequalities,
(c) a heuristic-based reduction of the model size.

While item (a) is specific to the y-variables and Constraints (1) from M1, (b) and (c) can be applied to M1 and M2. For this reason, we will discuss all these methods using M1, but also briefly point out how they might need to be modified for M2. For the sake of completeness, we again point out that the specification of a starting point for the ILP solver, which is implicitly associated with (c), can of course be used for all three existing models.

3.1 Reduction (a)

We first observe that after having applied Reduction (R1), Constraints (1) from M1 only need to be formulated for the time steps $t \in T(k)$ relevant on server $k \in K$. However, we do not require the whole inequality chain

$$y_{tk} \leq \sum_{i \in I_t : (i,k) \in \Delta} c_i x_{ik} \leq y_{tk} C$$

for each $t \in T(k)$, since either of the two parts has a very specific purpose. To be more precise, the left hand side is only important to perceive the deactivation of a server, while the right hand side helps to recognize an active server. Hence, it suffices to state the first inequality for ending times $t \in T_E(k) = \bigcup_{i \geq k} \{e_i\}$ (on server k) only, whereas the second inequality needs to be formulated just for the server-dependent starting times $t \in T_S(k) = \bigcup_{i \geq k} \{s_i\}$. In the latter case, we can even go a step further by noting that only the non-dominated starting times $T_S^{nd}(k)$ have to be considered. Observe that a starting time t_1 is called *dominated* if it is directly followed by another starting time t_2 (so that every job that is active at t_1 is still active at t_2), see also [10,17]. After we have accordingly broken the chain of inequalities into two parts, we can strengthen the first part by moving from $y_{tk} \leq \sum_{i \in I_t : (i,k) \in \Delta} c_i x_{ik}$ to $y_{tk} \leq \sum_{i \in I_t : (i,k) \in \Delta} x_{ik}$. Obviously, the latter better conveys the important message that an empty server cannot be active and an active server needs to have at least one item running on it. Overall, Reduction (a) helps to save some redundant conditions appearing in (1), while other conditions of that type are even tightened.

3.2 Reduction (b)

One of the major advantages of Model M3 from [17] consisted of the fact that the set of variables to be considered was very small due to the elimination of *forbidden item pairs*. By that, we mean that any pair of jobs appearing in $F := \{(i, j) \mid [s_i, e_i) \cap [s_j, e_j) \neq \emptyset, c_i + c_j > C\}$ cannot be executed on the same server. While this could be incorporated directly into the model generation of M3 due to the job-to-job coupling, additional valid inequalities are needed for M1 and M2 to avoid such pairs. However, the naive way of just demanding $x_{ik} + x_{jk} \leq z_k$ for any $k \in K$ and any $(i, j) \in F$ with (i, k), $(j, k) \in \Delta$ normally leads to a very large amount of further constraints (up to $\mathcal{O}(n^3)$ of them in the worst case). For this reason, here we propose a more sophisticated strategy that produces fewer and at the same time (partly) better such inequalities. To this end, note that for a fixed server k it would be sufficient to collect (a reasonable subset of) the maximal cliques of the *incompatibility graph* $\mathcal{G}(k) = (I(k), F)$ formed by the relation F on the set $I(k) := \{i \in I \mid i \geq k\}$ of items that can be assigned to server k. Then, for any such clique $\mathcal{C}$ (related to server k) with $|\mathcal{C}| \geq 2$, the condition

$$\sum_{i \in \mathcal{C}} x_{ik} \leq z_k \tag{17}$$

has to be added. For M1, we recommend to replace z_k on the right hand side of this inequality by y_{tk}, where t is the last starting time of the items from $\mathcal{C}$ (that is, one specific instant of time where all these items are active). Thanks to the coupling conditions (4) from M1, this will impose an even stronger constraint.
To find the maximal cliques, we propose the following strategy, starting with $k = 1$ (so that $I(k) = I$ holds):

(i) Find the maximal cliques of the subgraph (of $\mathcal{G}(1)$) formed only by the items having $c_i > C/2$.
(ii) For any fixed item $i \in I$ with $c_i \leq C/2$: consider the subgraph formed by the items j with $c_j > C - c_i$ that are adjacent to i in $\mathcal{G}(1)$. When we add $\{i\}$ to a maximal clique of this subgraph, then we end up with a maximal clique of $\mathcal{G}(1)$.

By these two steps, we will find all maximal cliques of $\mathcal{G}(1)$ thanks to the following result:

Lemma 1 *Let $k \in K$ be fixed. Then, a maximal clique of $\mathcal{G}(k)$ can have at most one item of size $\leq C/2$.*

Proof Indeed, if there were two such items in the clique, we would not have an edge between them which gives the contradiction. □

For any remaining server $k \geq 2$, the maximal cliques of $\mathcal{G}(k)$ (having size $|\mathcal{C}| \geq 2$) can be iteratively obtained from the information of the previous step $k - 1$ by deleting the item $i = k - 1$ from any maximal clique of $\mathcal{G}(k - 1)$ and applying cardinality and dominance tests to the obtained subsets of items.

3.3 Reduction (c)

The techniques presented so far (in Sects. 2 and 3) have not yet contributed to the reduction of the quantity $|K|$, which has a decisive influence on the model size as, for instance, the index k appears in any of the four variable types of M1. To deal with this challenge, which was already established as item (iii) in Sect. 2, appropriate heuristic information can be applied. However, from a theoretical point of view, the latter has only been successfully achieved for very small choices of γ:

Theorem 1 ([3]) *Let $\gamma \leq 1/n$. Then, the number of servers required for the TBPP-FU is equal to the number of servers in an optimal solution to the TBPP.*

The previous result allows for either computing the optimal size of K beforehand by solving a somewhat easier auxiliary problem (that is, the TBPP) or to at least limit the number of servers to any value obtained by a heuristic solution. However, as also reported in [3, Example 2.2], this result does not hold for larger choices of γ, so that reducing the size of K cannot be performed in the majority of the cases. To tackle this issue the following result presents an easy way of using heuristic information in the general case, too.

Theorem 2 *Let z_{heu} be the objective value obtained by any heuristic for the TBPP-FU. Then, the number of servers required in an optimal solution is at most $k^\star := \lfloor z_{heu}/(1+\gamma) \rfloor$.*

Proof If the claim was wrong we would need at least $k^\star + 1$ servers in an optimal solution. Since every server is switched-on at least once, this would lead to an objective value of at least

$$(k^\star + 1) + \gamma \cdot (k^\star + 1) = (1+\gamma) \cdot (k^\star + 1) > z_{heu}$$

giving the contradiction because the heuristic would have to be better than the optimal solution. □

This result allows us to replace the set $K = \{1, \dots, n\}$ at all positions in M1 and M2 with an appropriately defined and greatly reduced set $K^\star := \{1, \dots, k^\star\}$. Moreover, we recommend to pass the heuristic solution to the solver to give it a warm start. For our investigations, we will use the constructive look-ahead heuristic (CLH) described in [3, Sect. 3], but in a slightly more exploratory way. Before explaining the precise meaning of this intention, let us briefly collect the main idea of that heuristic: As stated in Sect. 2, we start by an item list ordered with respect to non-decreasing starting times s_i (where ties are broken in an arbitrary way) and process the items one by one. Moreover, we require a *look-ahead parameter* $q \in \mathbb{N}$ indicating the number of future items to be taken into account when making the current decision. In a specific iteration, we consider a fixed item $i \in I$ and assign it to every open server that is able to accommodate it, and (as another alternative) also to a new empty server. By that, we obtain various different assignments $\mathcal{A}_1, \dots, \mathcal{A}_p$. Now, we add the next q items to any of these assignments in a best-fit fashion, leading to the extended assignments $\widetilde{\mathcal{A}}_1, \dots, \widetilde{\mathcal{A}}_p$. Finally, we compute the corresponding objective values (i.e., the weighted sum of servers and fire-ups) and place item i to that bin whose extended assignment led to the lowest objective value.

Since in [3] the parameter $q = 3$ was used without compelling justification, we will first preface our actual test calculations in the next section with a somewhat more detailed consideration of the CLH algorithm.

4 Computational tests

4.1 Data sets and methodology

For our numerical calculations, we coded the above approaches in Python (version 3.9.2) and used its Gurobi (version 9.1.1) interface to solve the resulting ILP formulations with default settings and a time limit of 30 min per instance. All the experiments were run on an AMD A10-5800K processor with 16 GB RAM, that is, the same hardware as in [17]. Due to its novelty, the TBPP-FU has not yet been able to leave a large scientific footprint in the relevant literature, so that only one set of benchmark instances has been specifically designed for the problem under consideration, see [3]. In that publication, the authors propose 160 instances formed by 32 groups of five instances each, all sharing the same capacity $C = 100$. Apart from that, any group is determined by four indicators:

- number of items: $n \in \{50, 100, 150, 200\}$,
- time horizon: dense or relaxed (indicated by a maximum starting time $\bar{s} \in \{n, 1.2n\}$),
- job duration $d_i := e_i - s_i$: short or long (represented by 'd_S' and 'd_L' and indicated by uniformly distributed integers $d_i \in [10, 30]$ and $d_i \in [20, 60]$, respectively)
- resource consumption c_i: low or high (represented by 'c_L' and 'c_H' and indicated by uniformly distributed integers $c_i \in [25, 50]$ and $c_i \in [25, 75]$, respectively).

Even though the total number of instances appears to be relatively small, they can be considered very suitable for numerical test calculations due to their difficulty, especially because only 63 of them could be solved in the original publication [3]. Also the improvements discussed in [17] could increase this number to only just over 50% (that is, 85 out of 160), so that their solution still represents a serious challenge from today's point of view.

In the following discussions, the improved versions of the compact models from [17] will be referred to as M1*, M2*, and M3*. To reiterate, beyond providing a heuristic starting point, M1* and M2* differ from their previous versions by applying the techniques proposed in Sect. 3. For model M3, we note that based on the impressions of some internal preliminary test calculations, the final implementation in [17] sometimes did contain only a subset of the valid inequalities of type $x_{ik} \leq x_{kk}$, but unfortunately this was not sufficiently clearly stated in the text itself. Although this approach suggested slight performance advantages from the point of view of that time, (in the meantime) these expectations have not been generally confirmed with respect to the whole benchmark set tested here. For this reason, in contrast to [17], here we use the complete set of constraints of the above type (and, of course, the same heuristic starting point as for the other models) to define M3*.

4.2 Computational results for CLH

In a first experiment, we study the influence of the look-ahead parameter q on the performance of the CLH approach from [3]. For this purpose, we exemplarily consider the instances with $n \in \{100, 200\}$ items and refer to the results in Table 1. Based on this data, one can see the rough trend that a deeper look into the future (that is, a larger value of q) typically leads to a reduction in the heuristic value. However, this is by no means a strictly monotonous relationship, because for two different choices of q the obtained assignments will be considerably different, in general. Consequently, although the tabulated data give a very consistent picture in that large values of q are to be preferred, there is no single universally best choice of that parameter.

Remark 2 To better evaluate these data, column LB in Table 1 contains the average rounded-up LP values of M1*, i.e., a lower bound for the integer optimal value. By that, we see for instance that the average difference between the heuristic and the optimal value is bounded by roughly 27% for the hard instances with $n = 200$ items, but for a few constellations (especially with c_H) it is also (considerably) larger since it is much harder to obtain a dense heuristic packing in these cases.

As for the computational efforts, it is important to note that all the heuristic values can be determined very quickly, meaning that for many scenarios (n, q) the heuristic solution is available in less than 1 s. Even checking all the look-ahead parameters q (mentioned in Table 1) for an instance with $n = 200$ jobs and then deciding on the best result (see column 'best' in Table 1) takes only about 17 s on average, which is quite acceptable when measured against the time limit of 30 min permitted for the exact solution of these rather challenging instances. For this reason, and considering that our intention is to present a preferably maximally reduced compact formulation, we will always choose the best heuristic value to define $k^\star$ (that is, the number of initialized servers) appearing in Theorem 2. We note, however, that one could alternatively agree on a compromise between computational effort and quality of the heuristic solution and always use a fixed value of the look-ahead parameter (say $q = 20$), since already this leads to a significant improvement (e.g., on average about 15% better heuristic values for $n = 200$) compared to the relatively arbitrary choice of $q = 3$ from [3], without noticeably increasing the time required. Either way, as the cardinality of K strongly influences the number of variables and constraints, very powerful reductions in terms of the model size can be expected.

For the sake of exposition, we take a closer look at the associated numbers in Table 2. Due to space limitations, we again consider only the subset of instances that was also used in Table 1, but finally we also report the average results over all 160 instances in the last row of Table 2 to allow for a better overall picture. In addition, we also include the values of M3 (and M3* just having slightly more constraints for some instance groups), as this was the best formulation so far in terms of model size. Compared to that approach, we notice that the ideas presented in Sect. 3 lead to significant reductions of the integer programs associated with M1 and M2. To be more precise, while the

Table 1 Heuristic values for different choices of the look-ahead parameter q

n	$\bar{s}$	d_i	c_i	1	2	3	5	10	20	n/4	n/2	n	Best	LB
100	100	d_S	c_L	30.0	30.2	29.2	29.4	27.6	27.4	26.8	**25.8**	27.0	25.6	22.4
			c_H	48.4	46.4	45.0	43.0	40.4	**39.2**	39.4	40.4	39.4	38.2	33.6
		d_L	c_L	40.6	40.4	41.0	39.8	38.4	**37.8**	38.4	38.0	**37.8**	37.4	34.4
			c_H	66.2	64.4	62.6	60.6	58.6	55.2	56.0	55.0	**54.4**	54.4	49.2
	120	d_S	c_L	28.6	27.8	26.6	25.6	24.2	24.2	24.8	**24.0**	25.0	23.2	20.0
			c_H	45.6	42.4	41.6	39.8	39.0	**38.2**	38.4	39.6	39.4	37.2	27.2
		d_L	c_L	36.2	36.2	36.0	34.6	34.2	33.6	33.8	**32.8**	33.2	32.6	30.8
			c_H	60.0	59.2	57.6	55.2	53.0	55.2	51.6	51.6	**51.0**	50.2	44.4
Average				44.5	43.4	42.5	41.0	39.4	38.9	38.7	**38.4**	38.4	37.4	32.8
200	200	d_S	c_L	40.4	39.4	38.4	37.0	34.0	**32.8**	33.4	34.6	34.6	32.0	24.4
			c_H	69.4	67.8	65.4	60.4	55.8	**53.0**	53.8	57.6	58.4	51.8	31.6
		d_L	c_L	50.0	49.2	47.4	47.8	45.8	44.0	42.8	**42.4**	43.2	42.0	38.0
			c_H	89.8	86.0	83.6	78.8	72.6	69.8	68.0	**67.8**	69.4	66.6	53.6
	240	d_S	c_L	39.6	38.2	38.4	36.2	34.8	**30.6**	32.0	33.0	34.2	30.2	21.2
			c_H	76.6	71.6	67.6	64.4	60.4	**59.0**	59.2	61.4	61.2	58.0	30.6
		d_L	c_L	43.4	42.6	41.0	42.6	39.6	39.0	**37.4**	38.0	37.8	36.8	32.4
			c_H	84.0	81.2	77.6	71.8	67.6	**62.0**	62.6	62.8	65.6	61.2	46.0
Average				61.7	59.5	57.4	54.9	51.3	48.8	**48.6**	49.7	50.6	47.3	34.7

The best average value for each subset is printed in bold. The column 'best' refers to the average over the best value obtained per instance from the considered subset. The column 'LB' provides the (average) rounded-up LP value of M1* to enable a rough evaluation of the heuristic solution

Table 2 An overview of the numbers of variables n_{var} and constraints n_{con} (all of which represented in units of 10^3)

				n_{var}						n_{con}					
n	$\bar{s}$	d_i	c_i	M1	M1⋆	M2	M2⋆	M3	M3⋆	M1	M1⋆	M2	M2⋆	M3	M3⋆
100	100	d_S	c_L	13.5	2.9	8.4	**2.0**	5.2	5.2	24.0	3.6	13.8	**3.2**	13.4	13.4
			c_H	13.5	4.2	8.3	**2.8**	4.4	4.4	24.1	6.7	14.1	**6.2**	9.8	11.3
		d_L	c_L	14.4	4.1	8.4	**2.8**	5.2	5.2	26.6	4.6	13.7	**4.3**	13.4	13.4
			c_H	14.3	5.6	8.3	**3.8**	**3.8**	**3.8**	26.4	9.9	13.7	9.4	**9.0**	9.8
	120	d_S	c_L	14.2	2.8	8.6	**1.8**	5.2	5.2	25.5	3.4	13.9	**2.9**	13.6	13.6
			c_H	14.2	4.4	8.6	**2.9**	4.5	4.5	25.6	7.1	13.7	**6.2**	10.0	11.9
		d_L	c_L	14.7	3.8	8.6	**2.5**	5.2	5.2	27.2	4.4	13.8	**3.9**	13.6	13.6
			c_H	15.0	5.7	8.7	**3.7**	3.9	3.9	27.7	9.9	13.5	**9.1**	9.2	10.3
200	200	d_S	c_L	52.5	7.6	33.4	**5.0**	20.3	20.3	91.0	9.6	54.9	**8.3**	53.4	53.4
			c_H	51.8	11.8	33.1	**7.9**	18.7	18.7	89.8	20.9	56.2	**19.0**	41.0	48.9
		d_L	c_L	54.8	9.8	33.4	**6.5**	20.3	20.3	97.8	11.9	55.3	**10.5**	53.4	53.4
			c_H	53.3	14.7	32.9	**9.9**	17.2	17.2	94.3	28.6	54.9	**26.7**	39.2	45.1
	240	d_S	c_L	55.1	7.5	34.0	**4.8**	20.3	20.3	97.6	9.3	54.2	**7.7**	53.9	53.9
			c_H	55.9	14.4	34.4	**9.2**	18.9	18.9	99.3	23.4	54.2	**20.2**	42.3	50.7
		d_L	c_L	56.9	9.1	34.2	**5.9**	20.3	20.3	102.6	11.0	54.7	**9.2**	54.2	54.2
			c_H	56.5	14.7	34.1	**9.5**	17.7	17.7	101.7	26.3	55.0	**23.4**	40.7	47.3
Average (for $50 \leq n \leq 200$)				26.0	6.1	15.9	**4.0**	8.9	8.9	46.5	9.4	25.8	**8.4**	21.9	23.5

M1, M2, and M3 are the versions from the literature, see [17], whereas M1⋆, M2⋆, and M3⋆ contain the ideas specified in Sect. 4.1. The best average value per subset is printed in bold savings in the number of constraints is about 60% in both cases (compared to M3), in the case of the number of variables it ranges from about 32% (for M1*) to 45% (for M2*). These reductions become even more remarkable when referring only to the comparison of the literature version M1 (resp. M2) with the version M1* (resp. M2*) improved in the context of this work. On average, here we end up with roughly 75% fewer variables (in both cases) and, depending on the formulation, between 67% and 80% fewer constraints. While for a fixed number of items n and a fixed model we previously saw very little variation in the indicators n_{var} and n_{con} among the several groups of instances, we now observe that our reductions are particularly successful when short and/or low-resource jobs are considered (see d_S and c_L). On the one hand, these constellations tend to lead to particularly few interactions between the jobs and therefore allow for better heuristic solutions (typically leading to a small value of $k^\star$), which is also clearly supported by the results from Table 1. On the other hand, for the case c_H, the number of forbidden item combinations increases significantly, so that, for example, a sometimes substantial number of valid inequalities (see Reduction (b) in Sect. 3) must be added to the model.

4.3 Computational results for the compact models

In a next step, we focus on the performance of M1*, M2*, and M3* when addressing the exact solution of the benchmark instances. To this end, we tabulate the obtained results in Table 3 and compare them with the previous state of the literature (that is, M1, M2, and M3 from [17]). First, we note that the contributions from Sect. 3 (and also the warm start of the solver) helped to significantly increase the number of instances solved to optimality. More specifically, the modifications to M1 (M2, and M3) resulted in 18 (18, and 15) additional proven optimal solutions, so that all formulations now perform considerably better than their original versions from [17]. A table containing more information about which model was able to solve which instance can be found in the "Appendix" section.

Remark 3 Interestingly, in at least ten additional cases M2* already had the correct optimal value, but failed to prove the optimality within the given time limit. For M1* and M3*, these numbers resulted in 0 and 3 instances, respectively, see also Table 7 in the "Appendix" section.

Due to its small model size and the fact that, for example, the reduction related to K is particularly promising for large values of n, M3* is the best formulation for the very small instances with $n = 50$ items, but constantly loses this leading position for larger instances (especially in comparison with M1*). Overall, it is noticeable in Table 3 that the performance of M1*, M2*, and M3* has improved, especially for many instances from the constellation (d_S, c_L), which further supports the observations made in Table 2 that the reductions (of M1 and M2) are particularly strong for these cases. However, the ideas from Sect. 3 not only contribute to an overall improvement in the number of optimally solved instances, but (in many cases) also to considerably lower computation times required. On the one hand, this can be seen from the average values in Table 3, but it becomes somewhat more evident if we look at the percentage of optimally

Table 3 Number 'opt' of instances solved to optimality (from a total of five instances per subset), and average solution times t in seconds

				M1		M1*		M2		M2*		M3		M3*	
n	$\bar{s}$	d_i	c_i	t	opt	t	opt	t	opt	t	opt	t	opt	t	opt
50	50	d_S	c_L	7.4	(5)	4.1	(5)	6.3	(5)	8.4	(5)	**3.6**	**(5)**	4.9	(5)
			c_H	8.6	(5)	2.8	(5)	2.8	(5)	**1.2**	**(5)**	3.7	(5)	1.2	(5)
		d_L	c_L	367.8	(4)	360.5	(4)	71.9	(5)	361.1	(4)	**17.0**	**(5)**	18.3	(5)
			c_H	12.3	(5)	0.3	(5)	5.0	(5)	1.1	(5)	**0.2**	**(5)**	**0.2**	**(5)**
	60	d_S	c_L	14.0	(5)	0.6	(5)	3.3	(5)	**1.5**	**(5)**	4.6	(5)	2.5	(5)
			c_H	37.8	(5)	14.9	(5)	363.1	(4)	3.7	(5)	3.2	(5)	**2.6**	**(5)**
		d_L	c_L	251.2	(5)	46.0	(5)	5.6	(5)	4.7	(5)	1.1	(5)	**0.9**	**(5)**
			c_H	5.7	(5)	0.4	(5)	2.2	(5)	0.6	(5)	0.3	(5)	**0.2**	**(5)**
Average (Sum)				88.1	(39)	53.7	(39)	57.5	(39)	47.8	(39)	4.2	(40)	**3.9**	**(40)**
100	100	d_S	c_L	22.1	(5)	1.3	(5)	4.2	(5)	**0.4**	**(5)**	46.8	(5)	10.0	(5)
			c_H	738.6	(4)	**321.1**	**(5)**	956.7	(3)	728.0	(3)	1128.7	(3)	448.0	(4)
		d_L	c_L	1457.9	(1)	1469.9	(1)	1800.0	(0)	**1443.4**	**(1)**	1447.4	(1)	1445.4	(1)
			c_H	1500.4	(1)	**1440.2**	**(1)**	1555.6	(1)	**1440.2**	**(1)**	1443.4	(1)	1440.5	(1)
	120	d_S	c_L	91.2	(5)	**6.5**	**(5)**	152.7	(5)	16.6	(5)	381.8	(4)	22.2	(5)
			c_H	1042.8	(3)	**754.6**	**(4)**	1800.0	(0)	1800.0	(0)	1800.0	(0)	1515.1	(1)
		d_L	c_L	850.4	(3)	**373.8**	**(4)**	803.8	(3)	447.7	(4)	635.2	(4)	731.0	(3)
			c_H	1481.0	(1)	787.3	(3)	1329.9	(2)	1086.3	(2)	733.3	(3)	**727.6**	**(3)**
Average (Sum)				898.0	(23)	**644.4**	**(28)**	1050.4	(19)	870.3	(21)	952.1	(21)	792.5	(23)

Table 3 continued

				M1		M1⋆		M2		M2⋆		M3		M3⋆	
n	$\bar{s}$	d_i	c_i	t	opt	t	opt	t	opt	t	opt	t	opt	t	opt
150	150	d_S	c_L	462.4	(4)	23.9	(5)	404.5	(4)	**23.4**	**(5)**	601.1	(4)	402.3	(5)
			c_H	1800.0	(0)	**1729.0**	**(1)**	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
		d_L	c_L	1502.3	(1)	1247.5	(2)	1800.0	(0)	988.4	(3)	1726.6	(1)	**882.9**	**(3)**
			c_H	1517.8	(1)	1456.7	(1)	1800.0	(0)	**1453.1**	**(1)**	1628.7	(1)	1468.0	(1)
	180	d_S	c_L	93.7	(5)	22.3	(5)	377.7	(4)	**16.9**	(5)	634.1	(4)	191.0	(5)
			c_H	1682.6	(2)	**1301.7**	**(3)**	1800.0	(0)	1600.3	(1)	1800.0	(0)	1604.2	(1)
		d_L	c_L	1800.0	(0)	886.3	(3)	1620.0	(1)	**753.6**	**(3)**	1800.0	(0)	1141.3	(2)
			c_H	1800.0	(0)	1271.6	(2)	1649.1	(1)	1293.1	(2)	1797.6	(1)	**1246.3**	**(2)**
Average (Sum)				1332.3	(13)	**992.4**	**(22)**	1406.4	(10)	991.1	(20)	1473.5	(11)	1092.0	(19)
200	200	d_S	c_L	582.8	(4)	385.0	(4)	730.5	(3)	**371.5**	**(4)**	1595.9	(2)	752.2	(4)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
		d_L	c_L	1574.1	(1)	1536.1	(1)	1800.0	(0)	**1122.6**	**(2)**	1800.0	(0)	1534.3	(1)
			c_H	1800.0	(0)	**1293.7**	**(2)**	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
	240	d_S	c_L	220.1	(5)	**35.2**	**(5)**	1083.7	(2)	728.9	(3)	1144.1	(2)	1105.4	(2)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
		d_L	c_L	1800.0	(0)	1109.3	(2)	1800.0	(0)	**1096.7**	**(2)**	1800.0	(0)	1489.2	(2)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
Average (Sum)				1422.1	(10)	**1219.9**	**(14)**	1576.8	(5)	1315.0	(11)	1692.5	(4)	1510.1	(9)
Total: Average (Sum)				935.2	(85)	**727.6**	**(103)**	1022.8	(73)	806.0	(91)	1030.6	(76)	849.6	(91)
Average Exit gap [%]				20.8		**3.9**		9.5		5.2		9.5		4.9	

For the sake of completeness, also the average exit gaps (computed based on all 160 instances) are displayed. We use bold numbers to indicate the best formulation per subset

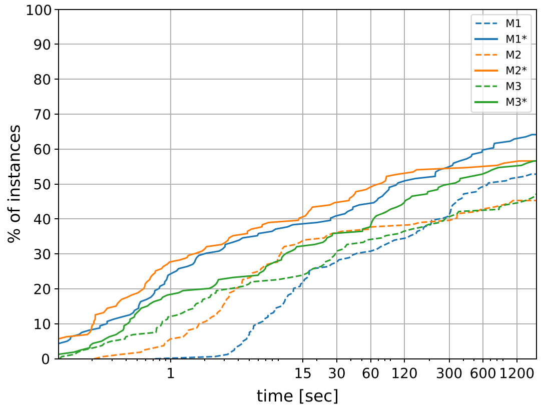

Fig. 2 Temporal development of the number of instances solved to optimality by the various formulations

solved instances over time, see Fig. 2. Therein, it is clearly visible that at any point in time a fixed updated formulation dominates its corresponding original model from [17]. Moreover, given the additional improvements from Sect. 3, either M1* or M2* always possesses the most convincing performance. The generally smaller model size of M2* causes it to dominate M1* within the first 2 min, while in the long term M3* offers similar and M1* even better numerical results. We attribute the latter to the fact that the numerous coupling conditions in M1* (i.e., the implications inherent to Constraints (3)–(5)) have a large effect in the deeper layers of the branch-and-bound tree, since already the specification of a small set of variables actually fixes a much larger set of variables to integer values. Moreover, the methods contained in Reduction (a), and the fact that the valid cuts from Reduction (b) can be formulated in a stronger way may also have contributed to the slightly better overall performance of M1*.

While these considerations refer only to the successfully solved instances, the exit gaps provided in the last row of Table 3 also indicate the improvements with respect to all instances. Roughly speaking, all compact models were able to reduce their exit gap by at least 45%, with M1* standing out here with a reduction of more than 80%. This observation is partly due to the fact that the specification of a starting point now generally leads to reasonable approximate solutions even for very difficult instances.

Remark 4 To gain a rough insight into which of the presented reductions have which effect, we exemplarily solved again the more difficult half of the instances (those with $n \geq 150$) by different variants of M1. More precisely, we start with the version from [17] and then gradually add the methods from Sect. 3, see Table 4. Note that for Reduction (c) we distinguish between the mere heuristic-based reduction (called "M1* (cold)") and the additional use of the feasible solution as a starting point for the ILP solver.

In terms of model size, one can clearly see that Reduction (c) makes the largest contribution, reducing both variable and constraint numbers very significantly. However, also Reduction (a) helps to make a remarkable improvement, especially by already removing roughly 40% of all constraints. In contrast, as expected, adding valid cuts (i.e., applying Reduction (b)) again leads to an increase (of 22%) in constraints, but this still results in better overall performance, especially for $n = 150$. With respect to

Table 4 Some key indicators summarizing the effects of the step-by-step reduction for $n = 150$ (upper half, i.e., rows 1-6) and $n = 200$ (lower half, i.e., rows 7-12)

Indicator	Units	M1	M1 + (a)	M1 + (a) + (b)	M1* (cold)	M1*
n_{var}	10^3	31.3	28.7	28.7	7.4	7.4
n_{con}	10^3	56.1	34.1	41.5	11.7	11.7
n_{nz}	10^3	613.0	317.0	408.5	130.9	130.9
z	–	61.7	57.5	46.6	38.0	37.4
opt	–	13	15	17	19	22
t	s	1332.3	1228.3	1174.8	1084.6	992.4
n_{var}	10^3	54.6	51.1	51.1	11.2	11.2
n_{con}	10^3	96.8	61.1	74.5	17.6	17.6
n_{nz}	10^3	1114.2	607.4	770.0	199.3	199.3
z	–	179.1	95.4	91.6	44.3	40.1
opt	–	10	10	10	11	14
t	s	1422.1	1376.0	1376.3	1360.1	1219.9

In addition to the notation already used before, n_{nz} represents the number of nonzero elements appearing in the constraint matrix of the optimization problems

the number of instances solved to proven optimality and the solution times required, it can be observed that for $n = 150$ each individual method has an approximately equal contribution. For the even more difficult instances with $n = 200$ items, Reductions (a) and (b) do particularly lead to significantly better objective function values (in total, we see a reduction of almost 50% from M1 to M1+(a)+(b)), but the optimality for additional instances can only be witnessed after having added Reduction (c). Hence, from the point of view of model performance, Reduction (c) as a whole (i.e, using heuristic information plus warm start) is typically slightly superior to the other two individual techniques. Some more detailed results can also be found in Table 8 in the "Appendix" section.

Moreover, Fig. 3 additionally gives an overview of the average objective values (again only for the 80 harder instances with $n \geq 150$) over time. Besides the obvious and substantial improvements of M1*, M2*, and M3* (over the original versions), we highlight the very good performance of M2* (see Fig. 4 for an enlarged section) at almost all instants of time, which we again particularly attribute to the much smaller model size. Although Fig. 4 might suggest this, M2* is nevertheless not better than M1* or M3* for every single instance, see also Table 5.

We observe that the comparison between M1* and M2* ends in a draw here, with 16 wins for each of the models, while both of the previously mentioned formulations dominate the M3* model in terms of the objective function value found much more often than they are defeated by it. From a general point of view, it can be said that the advantages of M1* lie in particular in its ability to obtain proven optimal solutions, while M2* is able to (on average) produce slightly better feasible points even for challenging instances of the TBPP-FU. The high quality of the approximate solutions obtained from M1* and M2* is also confirmed by the fact that the best (rounded-up)

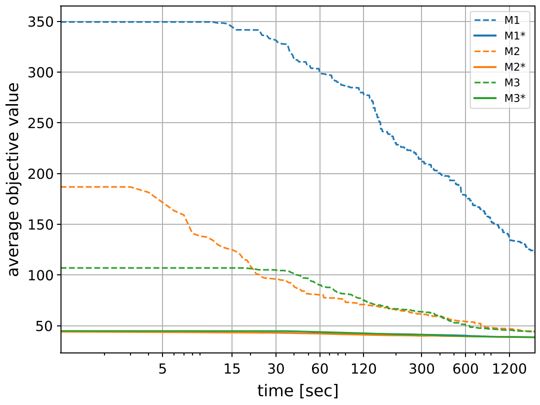

Fig. 3 Temporal development of the average objective value for $n \geq 150$ by the various formulations. To better distinguish the new models, which are very close to each other, we refer to Fig. 4

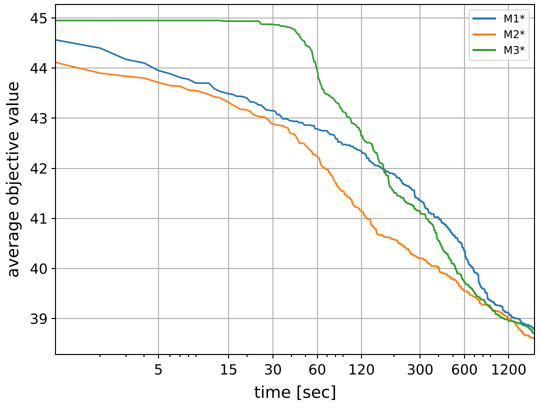

Fig. 4 An enlarged section from Fig. 3 to better compare M1*, M2*, and M3*

LP bound (over all models) for the instances considered in Fig. 4 averaged 34.55. The fact that M3*, in the light of Table 5, now tends to perform worst in the comparison of the three formulations is mainly due to the fact that, according to Table 2, it has lost its former leading position (in terms of model size) to the other two formulations as a result of the very powerful improvements from Sect. 3.

Overall, it can be concluded that the methods presented in Sect. 3 (and the warm start of the solver) not only substantially improve the performance of the models individually, but also result in the advantageous features of the M3-type approaches (listed in [17]) now being barely discernible in most of the numerical comparisons. As a consequence, also M3* is outperformed in many respects by M1* and M2*.

4.4 An outlook: valid cuts from lot sizing

In the literature there are many problems of operations research which, similar to the TBPP-FU, assign additional costs to the start-up of a machine, see for example the uncapacitated lot sizing problem [21]. For that problem, classes of valid inequalities are also known, whose applicability and usefulness for the TBPP-FU we want to briefly discuss here as a conclusion of our considerations. In particular, this is also

Table 5 Comparison between the three improved formulations

	$M1^\star$	$M2^\star$	$M3^\star$
$M1^\star$	–	16	25
$M2^\star$	16	–	26
$M3^\star$	15	15	–

An integer number $x \in \mathbb{N}$ at position (i, j) in the table means, that formulation i was strictly better (i.e., it found a better feasible point) than j in x cases

to emphasize that the improvements in compact models achieved in this article have indeed reached a certain plateau level and, as a consequence, that obvious ideas from adjacent fields do not easily lead to further numerical advantages. For the sake of exposition, in a final experiment, let us therefore add the following types of constraints to $M1^\star$

$$\begin{aligned} w_{tk} &\leq y_{tk}, \\ w_{tk} &\leq 1 - y_{t-1,k}, \\ lb_t &\leq \sum_{k \in K^\star} y_{tk}. \end{aligned}$$

The first two sets are directly taken from [21], establish an additional w-y-coupling of the variables appearing in $M1^\star$, and can be added for all $k \in K^\star$ and $t \in T_S(k)$. In particular, these restrictions include that a server that is active at a given instant of time cannot be activated at the following point in time. The third set of conditions involves a lower bound $lb_t \in \mathbb{N}$ which is defined as the optimal value of a bin packing problem containing precisely the subset I_t of jobs being available at time $t \in T_S$. By that, we make sure that sufficiently many servers are active at every time instant to accommodate the items that are executed at that moment.

Remark 5 Since all these valid cuts explicitly require the presence of the y-variables, they cannot be applied to $M2^\star$ or $M3^\star$.

Let us refer to the model containing all these new valid inequalities by $M1^{\star\star}$. Then, for the benchmark set and hardware specified in Sect. 4.1, we obtain the average numbers collected in Table 6. In terms of model size, we note that variables remain untouched, while there is an obvious increase in the number n_{con} of constraints and also in the number n_{nz} of non-zero elements in the system matrix. In contrast, the LP bound z_{LP} at the root node improves only negligibly. In our observations, these opposing effects nevertheless lead to a marginal improvement of the model performance overall. As can be seen from the exit gap, slightly better feasible points are found on average, so that in the end exactly one more instance could be optimally solved. We attribute this in particular to the fact that although the additional inequalities do not necessarily contribute to raising the continuous bound in the root node, they do help to keep the resulting branch-and-bound trees (over the entire time period) somewhat smaller, see also Fig. 5.

Table 6 A brief overview of the numerical comparison between M1* and M1**

Indicator	Units	M1*	M1**	Diff
n_{var}	10^3	6.1	6.1	±0%
n_{con}	10^3	9.4	12.6	+34%
n_{nz}	10^3	102.1	110.0	+7.7%
z_{LP}	–	32.7	32.8	+0.3%
Exit gap	%	3.9	3.4	−12.8%
opt	–	103	104	+1%
t	s	727.6	705.5	−3%

The column 'Diff' measures the percentage deviation of the new value from the previous one. Please note that positive and negative deviations may have a separate meaning depending on the criterion

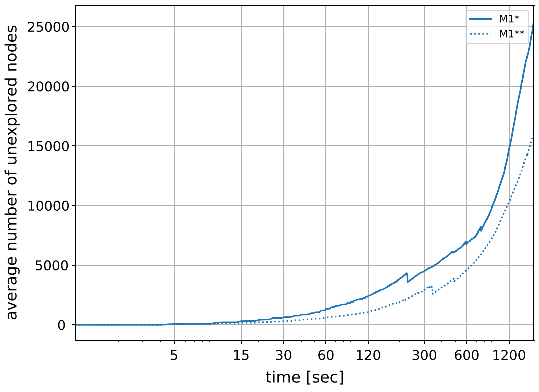

Fig. 5 The average number of unexplored nodes (for M1* and M1**) over time. The average is built based on all 160 instances

Finally, however, we would like to point out that despite the same software and hardware, the solution process applied by the Gurobi solver is subject to a high degree of randomization and thus a comparison of both sets of experiments (with such a small difference) is difficult. In particular, we do not want to claim that the small performance deviations between the two variants are an actual advancement of the model itself, since they could have been caused by other effects. Even though we will refer to M1* as state-of-the-art for these reasons, it was important to us to at least briefly discuss this alternative variant in the context of an outlook, since one or the other version could prove to be more advantageous for concrete problem instances from practice or other (future) benchmark sets.

5 Conclusions

In this article, we dealt with the temporal bin packing problem with fire-ups, a relatively new decision making problem in operations research typically leading to integer models of challenging size. Even though some fundamental methods for obtaining more tractable formulations have already been described in the recent literature, these investigations do not yet turn out to be "complete" upon closer inspection, especially because the incorporation of heuristic information has so far only been possible for a few special cases. Therefore, the contributions of this article were aimed in particular at three methods to improve existing ILP models: “optimizing” the set of constraints (by removing redundancies and tightening some inequalities), adding valid inequalities, and reducing the number of servers to be considered (thus considerably decreasing the overall model size). Based on numerical computations, the positive effects of the new techniques (together with the warm start of the solver) could be manifested. We underline not only the fact that, as a result of the improvements, each model was able to solve at least 15 additional instances (compared to its previous version from [17]) of the challenging benchmark set from [3], but also highlight that, in particular, the optimal solution of 14 instances (twelve by M1*, nine by M2*, seven by M3*, and six by all three models) was obtained for the first time. Now that the investigation of assignment models for the TBPP-FU is somewhat “complete”, future research should focus in particular on flow-based models or branch-and-price approaches. Moreover, theoretical results dealing with the worst-case performance of heuristics for the TBPP-FU have not yet been addressed at all in the literature. From a practical point of view, also generalizations of the problem considered here could be explored, in which, for example, the time interval $[s_i, e_i)$ of a job can be shifted slightly, which is then associated with penalty costs (for early or late execution). An idea related to this in a certain sense has already been mentioned in the concluding parts of [10] for the TBPP.

Funding Open Access funding enabled and organized by Projekt DEAL.

Availability of data and material The instances used in that paper were originally designed in [3] and can be found online, see https://github.com/sibirbil/TemporalBinPacking.

Declarations

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

Code availability The instances were solved by the commercial software Gurobi. The underlying implementation of the models in Python can be found in https://github.com/wotzlaff/tbpp-cf2.

A Further numerical results

See Tables 7 and 8.

Table 7 Detailed numerical results for every instance (numbered from 1 to 5 for any parameter constellation) of the considered benchmark set

				z_{best}					M1⋆					M2⋆					M3⋆				
n	$\bar{s}$	d_i	c_i	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
50	50	d_S	c_L	18	20	18	22	20	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
			c_H	24	28	27	28	22	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
		d_L	c_L	32	30	28	30	30	S	S	S	S		S	S	S	S		S	S	S	S	S
			c_H	38	44	48	54	46	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
	60	d_S	c_L	16	16	18	16	18	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
			c_H	24	29	23	24	20	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
		d_L	c_L	28	32	30	30	28	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
			c_H	44	38	38	48	40	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
100	100	d_S	c_L	20	22	24	24	22	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
			c_H	41	35	33	33	32	S	S	S	S	S	S	S	V	V	S	S	S	S	V	S
		d_L	c_L	36	38	36	36	34	V	V	V	V	S	V	V	V	V	S	V	V	V	V	S
			c_H	46	58	48	46	56	V	V	S	V	V	V		S	V	V	V		S	V	V
	120	d_S	c_L	20	22	18	20	20	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
			c_H	35	28	32	27	31	S	S	S	V	S	V	V	V	V	V		S	V	V	V
		d_L	c_L	30	30	30	34	32	S	S	V	S	S	S	S	V	S	S	S		V	S	S
			c_H	42	52	42	48	44	V	S	S	V	S		V	S	V	S	V	S	S	V	S

Table 7 continued

				z_{best}					M1*					M2*					M3*				
n	$\bar{s}$	d_i	c_i	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
150	150	d_S	c_L	20	22	20	24	22	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
			c_H	40	46	41	35	35	S	V	V	V			V	V		V				V	
		d_L	c_L	38	42	40	40	38	S	V		V	S	S	V	S	V	S	S	V	S	V	S
			c_H	51	57	57	58	50		V		S		V			S				V	S	V
	180	d_S	c_L	20	18	18	20	22	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
			c_H	41	25	33	38	38		S		S	S	V	S	V	V	V		S			
		d_L	c_L	32	34	32	32	32	S	V	V	S	S	S	V	V	S	S	S	V	V		S
			c_H	51	48	58	55	48	S				S	S	V	V		S	S			V	S
200	200	d_S	c_L	24	24	29	22	24	S	S	V	S	S	S	S	V	S	S	S	S	V	S	S
			c_H	46	49	39	29	37		V	V	V		V			V	V					
		d_L	c_L	40	40	40	36	40	V	S	V		V	V	S	V	S	V	V	S	V		V
			c_H	57	58	50	59	65		S	S	V							V				V
	240	d_S	c_L	22	20	22	20	24	S	S	S	S	S	V	S	S	S			S		S	
			c_H	41	42	41	54	48	V	V			V		V	V						V	
		d_L	c_L	36	30	34	36	32	V	S	S	V	V	V	S	S	V	V	V	S	S	V	V
			c_H	53	43	56	54	53		V	V	V		V	V							V	V

The table contains the best objective value z_{best} found by any of the models M1*, M2*, and M3* as well as the information which approach led to that value (indicated by 'V'). Whenever, in addition, an instance was solved to proven optimality by a given formulation, we use the symbol 'S' (instead of 'V'). This table also supports the observations from Remark 3

Table 8 Detailed results for the 80 instances with $n \in \{150, 200\}$ solved by different variants of M1

				M1		M1+(a)		M1 + (a) + (b)		M1⋆ (cold)		M1⋆	
n	$\bar{s}$	d_i	c_i	t	opt	t	opt	t	opt	t	opt	t	opt
150	150	d_S	c_L	462.4	(4)	296.1	(5)	299.5	(5)	69.3	(5)	23.9	(5)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1729.0	(1)
		d_L	c_L	1502.3	(1)	1458.3	(1)	1458.7	(1)	1455.7	(1)	1247.5	(2)
			c_H	1517.8	(1)	1578.9	(1)	1349.4	(2)	1504.5	(1)	1456.7	(1)
	180	d_S	c_L	93.7	(5)	20.8	(5)	21.2	(5)	4.4	(5)	22.3	(5)
			c_H	1682.6	(2)	1509.9	(1)	1442.7	(2)	1200.4	(3)	1301.7	(3)
		d_L	c_L	1800.0	(0)	1448.8	(1)	1448.9	(1)	1108.5	(2)	886.3	(3)
			c_H	1800.0	(0)	1713.3	(1)	1578.1	(1)	1534.0	(2)	1271.6	(2)
Average (Sum)				1332.3	(13)	1228.3	(15)	1174.8	(17)	1084.6	(19)	992.4	(22)
200	200	d_S	c_L	582.8	(4)	430.9	(4)	431.9	(4)	393.1	(4)	385.0	(4)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
		d_L	c_L	1574.1	(1)	1492.8	(1)	1493.5	(1)	1659.7	(1)	1536.1	(1)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1583.5	(1)	1293.7	(2)
	240	d_S	c_L	220.1	(5)	84.6	(5)	85.2	(5)	44.5	(5)	35.2	(5)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
		d_L	c_L	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1109.3	(2)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
Average (Sum)				1422.1	(10)	1376.0	(10)	1376.3	(10)	1360.1	(11)	1219.9	(14)

We start with the original version from [17], which is then improved step-by-step with the methods presented in Sect. 3. Reduction (c) is split into two parts: at first, we just reduce the number of servers by using heuristic information, but we do not provide the feasible starting point (called "M1⋆ (cold)" in the table), then, in a second step, we make use of the warm start option (leading to M1⋆)

References

1. Andrae, A.S.G., Edler, T.: On global electricity usage of communication technology: trends to 2030. Challenges **6**(1), 117–157 (2015)
2. Arkin, E.M., Silverberg, E.B.: Scheduling jobs with fixed start and end times. Discret. Appl. Math. **18**(1), 1–8 (1987)
3. Aydin, N., Muter, I., Birbil, S.I.: Multi-objective temporal bin packing problem: an application in cloud computing. Comput. Oper. Res. **121**, Article 104959 (2020)
4. Bartlett, M., Frisch, A.M., Hamadi, Y., Miguel, I., Tarim, S., Unsworth, C.: The temporal knapsack problem and its solution. Lect. Notes Comput. Sci. **3524**, 34–48 (2005)
5. Caprara, A., Furini, F., Malaguti, E.: Uncommon Dantzig–Wolfe reformulation for the temporal knapsack problem. INFORMS J. Comput. **25**(3), 560–571 (2013)
6. Caprara, A., Furini, F., Malaguti, E., Traversi, E.: Solving the temporal knapsack problem via recursive Dantzig–Wolfe reformulation. Inf. Process. Lett. **116**(5), 379–386 (2016)
7. Caprara, A., Toth, P.: Lower bounds and algorithms for the 2-dimensional vector packing problem. Discret. Appl. Math. **111**(3), 231–262 (2001)
8. Clautiaux, F., Detienne, B., Guillot, G.: An iterative dynamic programming approach for the temporal knapsack problem. Eur. J. Oper. Res. **293**(2), 442–456 (2021)
9. de Cauwer, M., Mehta, D., O'Sullivan, B.: The temporal bin packing problem: an application to workload management in data centres. In: Proceedings of the 28th IEEE International Conference on Tools with Artificial Intelligence, pp. 157–164 (2016)
10. Dell'Amico, M., Furini, F., Iori, M.: A branch-and-price algorithm for the temporal bin packing problem. Comput. Oper. Res. **114**, Article 104825 (2020)
11. Delorme, M., Iori, M., Martello, S.: Bin packing and cutting stock problems: mathematical models and exact algorithms. Eur. J. Oper. Res. **255**, 1–20 (2016)
12. 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.: Architecture and advanced electronics pathways towards highly adaptive energy-efficient computing. Proc. IEEE **107**(1), 204–231 (2019)
13. Iori, M., de Lima, V.L., Martello, S., Miyazawa, F.K., Monaci, M.: Exact solution techniques for two-dimensional cutting and packing. Eur. J. Oper. Res. **289**(2), 399–415 (2021)
14. Jones, N.: How to stop data centres from gobbling up the world's electricity. Nature **561**, 163–166 (2018)
15. Kantorovich, L.V.: Mathematical methods of organising and planning production. Manag. Sci. **6**, 366–422 (1939 Russian, 1960 English)
16. Martello, S., Monaci, M., Vigo, D.: An exact approach to the strip-packing problem. INFORMS J. Comput. **15**(3), 310–319 (2003)
17. Martinovic, J., Strasdat, N., Selch, M.: Compact integer linear programming formulations for the temporal bin packing problem with fire-ups. Comput. Oper. Res. **132**, Article 105288 (2021)
18. Scheithauer, G.: Introduction to cutting and packing optimization—problems, modeling approaches, solution methods. In: International Series in Operations Research & Management Science, vol. 263. Springer (2018)
19. Spieksma, F.C.R.: A branch-and-bound algorithm for the two-dimensional vector packing problem. Comput. Oper. Res. **21**, 19–25 (1994)
20. Valério de Carvalho, J.M.: LP models for bin packing and cutting stock problems. Eur. J. Oper. Res. **141**(2), 253–273 (2002)
21. Wolsey, L.A.: Uncapacitated lot-sizing problems with start-up costs. Oper. Res. **37**(5), 741–747 (1989)