Technische Universität Dresden
Herausgeber: Der Rektor

Compact Integer Linear Programming Formulations for the Temporal Bin Packing Problem with Fire-Ups

John Martinovic, Nico Strasdat, Maximilian Selch

Preprint MATH-NM-03-2020

October 2020

Compact Integer Linear Programming Formulations for the Temporal Bin Packing Problem with Fire-Ups

J. Martinovic[a,*], N. Strasdat[a], M. Selch[a]

[a]*Institute of Numerical Mathematics, Technische Universität Dresden, Germany*

Abstract

In this article we examine a specific version of the temporal bin packing problem (TBPP) that occurs in job-to-server scheduling. The TBPP represents a generalization of the well-known bin packing problem (BPP) with respect to an additional time dimension, and it requires to find the minimum number of bins (servers) to accommodate a given list of items (jobs) at any instant of time. In addition to this goal, recent publications suggest to also incorporate the number fire-ups of the respective servers as an important factor in sustainable and energy-efficient overall operation. Addressing both these objectives by a weighted sum method typically increases the modeling complexity, thus leading to challenging ILP formulations, two of which have already been described in the literature. It has been shown that the parameter used to weight the two objectives strongly influences the applicability of heuristics and, therefore, the difficulty of the overall problem, so that only favorable choices can be handled so far. But also for these tailored scenarios, the current approaches already fail to compute an exact solution of many moderately-sized instances in reasonable times. For this reason, we propose various new preprocessing techniques to strengthen the LP bound and/or to reduce the numbers of variables or constraints appearing in the existing ILP formulations as well as one alternative modeling approach for the problem under consideration. Based on numerical tests with differently characterized sets of benchmark instances, the new and improved formulations are shown to lead (on average) to better performances (than compact state-of-the-art models) in terms of instances solved to optimality and computation times.

Keywords: Cutting and Packing, Temporal Bin Packing Problem, Fire-Ups, Integer Programming, Reduction Methods

1. Introduction

The *bin packing problem* (BPP) is a widely studied combinatorial optimization problem, and it has been carefully investigated in numerous scientific articles, see [28, Fig. 1] for an illustrated statistical overview or [29] for a general survey and further references. Given a bin capacity $C \in \mathbb{N}$ and a list of $n \in \mathbb{N}$ items, each being characterized by some integer size $c_i \leq C$, $i \in I := \{1, \dots, n\}$, the BPP aims to find the minimum number of bins required to accommodate all items without violating the capacity constraint of a single bin. Special interest arises from a wide variety of real-life applications in scheduling or logistics [7, 22], of which the closest connection is certainly with the *cutting stock problem* (CSP), see [29, 54, 57]. Over time, a large number of different exact modeling frameworks for this problem has been described in the literature. Starting with early publications on assignment-based integer linear programs (ILPs) [39], nowadays' more sophisticated approaches typically involve column-generation [33, 34, 58],

[*]Corresponding author

Email addresses: `john.martinovic@tu-dresden.de` (J. Martinovic), `nico.strasdat@tu-dresden.de` (N. Strasdat), `maximilian.selch@tu-dresden.de` (M. Selch)

pseudo-polynomial formulations [12, 27, 51, 57], or branch-and-bound based techniques [10, 56, 60, 61]. Due to the NP-hardness of the BPP [48], a significant body of works also deals with approximation algorithms and their mathematical properties [18, 19, 30, 37].

In this article, we examine a particular version of the *temporal bin packing problem* (TBPP). The TBPP adds a time dimension to the ordinary BPP, meaning that any item $i \in I$ is equipped with its item size c_i and a lifespan $[s_i, e_i)$, where $s_i \in \mathbb{Z}_+$ and $e_i \in \mathbb{Z}_+$ denote the start and end time, respectively, see Fig. 1 for an example. Typically, the aim is to find a feasible assignment of jobs to as few bins as possible, so that the capacity restriction of every bin is respected for any instant of time.

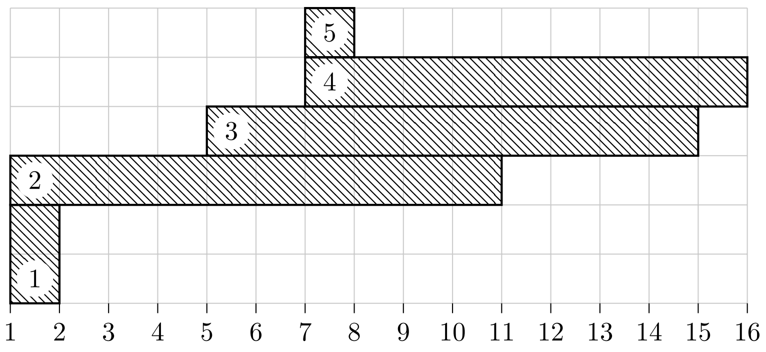


Figure 1: An examplary instance of the TBPP containing five items. The horizontal axis specifies the time instants, while the vertical grid measures the item sizes. (A corresponding solution will be illustrated in Fig. 3.)

The TBPP represents a relatively new field of research, whose mathematical origins can be motivated in at least three different ways:

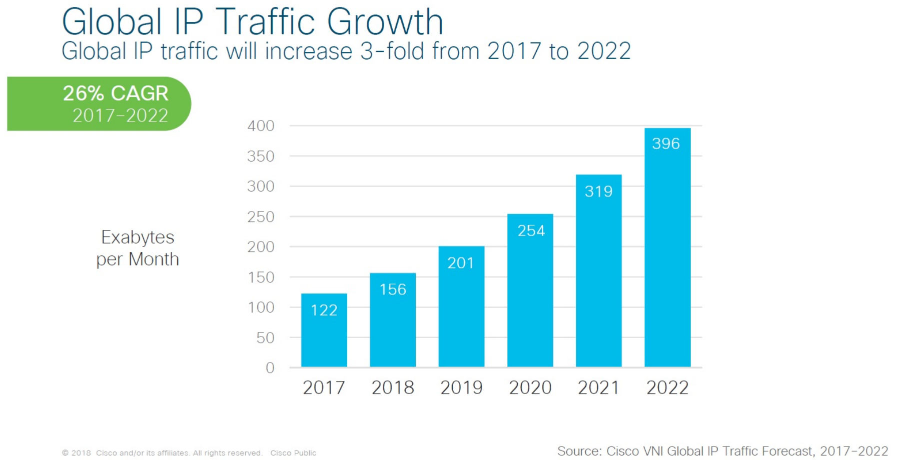
- The TBPP is a natural generalization of the *temporal knapsack problem* (TKP), which requires to select a subset of objects that will fit into a single bin of given capacity at any given time, while generating maximum profit. Leaving out the investigation of some special cases (like [3]), the general problem was introduced first in [9] to address resource allocation in high-performance computing, and it was solved by a decomposition method generating a tree of subproblems each of which being easier to handle (than the original problem). Moreover, the authors investigated the benefits of additional cuts and different strategies to select the next subproblem to be considered and obtained results that were (on average) competitive with the general-purpose solvers of that time. Meanwhile, the TKP has also been thoroughly studied from a mathematical point of view, the most significant contributions of which are especially due to [14, 15], where the authors applied a Dantzig-Wolfe reformulation and solved the problem under consideration within a branch-and-price algorithm.

- Alternatively, the TBPP can be modeled as a special case of a higher-dimensional *vector packing problem* (VPP), see [16, 35, 55] and references therein for a good overview on the general topic and some important solution techniques. More precisely, with $T := \bigcup_{i \in I} \{s_i, e_i\}$ denoting the instants of time to be considered, we could equip any item $i \in I$ with a time-dependent weight c_{it} equalling c_i if and only if $t \in [s_i, e_i)$ holds (otherwise, we set $c_{it} = 0$). This directly leads to a one-to-one correspondence between the TBPP and a VPP of dimension $|T|$, where every instant of time refers to a separate dimension. However, since this approach significantly increases the complexity of the initial problem, we cannot expect that this relationship can be used profitably here.

- To some extent, the TBPP also shares some similarities with classical two-dimensional packing problems, see [36, 43] for comprehensive survey articles. To be more precise, assigning rectangular-shaped items (here the rectangles consist of one geometric and one temporal dimension) to a minimum amount of "material" is somewhat reminiscent of the two-dimensional *strip packing problem* (SPP), see [21, 41, 47]. However, note there are two main differences between both formulations: In the TBPP, the items (i) have a fixed position with respect to the temporal dimension (whereas the placement for the SPP is much less restricted), and, more importantly, (ii) can make use of different portions of the bins at every instant of time (while, for the SPP, any item always has to be placed as a connected rectangle, meaning that it will not change the units covered on the vertical axis). We refer the interested reader to [26, Sect. 3] for a more detailed and illustrative description of these differences.

Due to its novelty, only a few works from the literature directly deal with the TBPP itself. More precisely, its first scientific investigation is based on an application in computer science trying to manage the workload consolidation in data centers [24]. Given the steadily increasing importance of cloud computing (see Fig. 2), the energy demand of large-scale data centers or server clusters is expected to grow considerably in the next decade [1, 2, 42], contributing to up to 13 percent of the worldwide energy consumption in 2030, see [38] for a general overview.

Figure 2: Predicted development of the data center IP traffic according to [8]. (The abbreviation CAGR refers to the compound annual growth rate.)



It is clear that managing this enormous traffic inevitably requires a very high number of active servers, most of which are usually underutilized (for fear of not being able to guarantee high availability at peak times), as several independent studies have shown [23, 46, 53]. Overall, this alarming development of energy consumption has caused deep concerns in industry and scientific communities [13, 31, 40, 45], so that addressing the problem examined here (that is, the TBPP) shall contribute to move towards balancing the supply of and the demand for computing resources as a key issue to obtain energy-efficient server consolidations. Of course, the TBPP (like any other mathematical model, too) can only reflect partial aspects of what is in reality a much more complex allocation problem. We would therefore like to inform the reader that there are many neighboring approaches that focus on other characteristic features (e.g., the uncertainty in terms of item sizes [20, 50, 52]), see [44] for a very thorough and well-structured overview.

From a mathematical point of view, the TBPP was extensively investigated in [26] for the first time. In that article, the authors present several methods to obtain lower and upper bounds, as well as two ILP formulations (one of which is based on a pattern structure, while the other uses classical assignment variables). Furthermore, some preprocessing techniques are described and connections or dominance relations between these approaches are manifested by theoretical results. Based on all these contributions, a branch-and-price algorithm is formed showing a very convincing computational performance for a large set of test instances.

While the previous paper only addressed the minimization of the number of required bins (or servers), another critical aspect of energy-efficient scheduling, that is the number of *fire-ups*[1], was very recently identified by the authors of [4]. To reiterate, in addition to the mere number of servers in use, the operating mode (in particular the switch-on and switch-off processes) is also of great relevance for the energy consumption of the overall cluster. Taking into account both objectives, the authors introduced two ILP formulations merging the two goals in a weighted-sum fashion, thus establishing a multiobjective version of the TBPP, hereinafter referred to as the *temporal bin packing problem with fire-ups* (TBPP-FU), see Fig. 3.

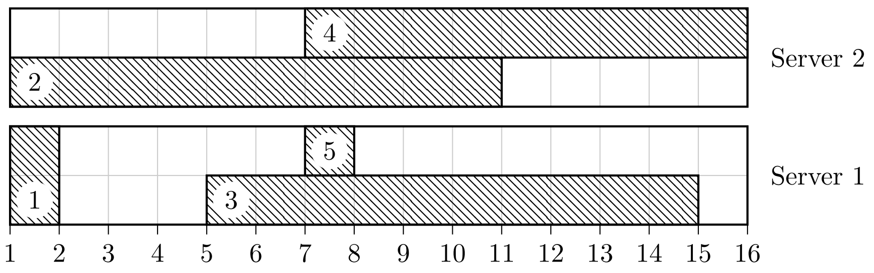


Figure 3: An optimal solution (with respect to both, the TBPP and the TBPP-FU) for the instance from Fig. 1. This assignment requires two bins (servers) and causes three fire-ups, leading to an objective value of 5 (in case of equal weights).

For both these modeling approaches, the quality of the LP bounds is studied and some preprocessing methods to obtain less complex integer programs are already discussed. By way of example, we mention that the information of an adapted material bound can be used to already preallocate some servers. Contrary to that, we would like to stress that the applicability of heuristics to also establish upper bounds for the total number of required servers strongly depends on the choice of the parameter $\gamma > 0$ used to weight the two objectives. More precisely, in [4, Sect. 2], it has been proved that for $\gamma \leq 1/n$ the number of servers in an optimal solution (of the TBPP-FU) is equal to the minimum number of servers needed to accommodate all jobs obtained by solving a traditional TBPP (i.e., without considering any fire-ups). In particular, this means that all the techniques to obtain upper bounds for the TBPP, see [26], can be used to reduce the set of possible servers in advance, so that much fewer feasible points (and much fewer symmetric solutions) have to be dealt with. Contrary to this, [4, Example 2.2] illustrates that such observations do not hold for larger values of γ. For this reason the authors of [4] only deal with the favorable case (that is, choosing $\gamma = 1/n$), which enables them to significantly simplify the ILP model by heuristic-based preprocessing techniques. The present work therefore aims in particular to cope with the general (and more difficult) case where bounding the number of required servers from above is not possible with the information provided by the ordinary TBPP. Although not being limited to this assumption, we will usually assume $\gamma = 1$ throughout the article, meaning that both objectives are weighted equally.

Altogether, after having repeated some important definitions and notations in Sect. 2, the present work shall foster theoretical approaches contributing to further increase the size of instances (of the TBPP-FU) that can be solved in a reasonable amount of time. More precisely, the main achievements (together with the structure and contents of this manuscript) are the following:

[1]A server that is currently not in use can be temporarily switched off, but it must be reactivated later in case it is used again. This process will be counted as a fire-up.

- We present several reduction techniques for the two state-of-the-art models from [4] and discuss their theoretical benefits ($\rightarrow$ Sect. 3).
- We introduce a new assignment model having significantly fewer variables than the formulations from [4] and adapt the previously established reduction strategies to this new approach ($\rightarrow$ Sect. 4).
- We theoretically show that there are no dominance relations between the (rounded-up) LP bounds of the three approaches ($\rightarrow$ Sect. 4).
- Based on extensive numerical tests, involving different benchmark sets, the computational benefits of our methods are verified ($\rightarrow$ Sect. 5).

2. Preliminaries and Assignment Models from the Literature

Let us consider a list of $n \in \mathbb{N}$ *items* (*jobs*), specified by an item *size* (*resource demand*) c_i and an *activity interval* (*lifespan*) $[s_i, e_i)$, $i \in I$, a sufficiently large number of homogeneous *bins* (*servers*) of *capacity* C, and a *weighting parameter* $\gamma > 0$ (as mentioned earlier, we will only use $\gamma = 1$ in the computational part). We will refer to s_i and e_i by the *starting time* and *ending time* (or *terminating time*), respectively. Without loss of generality, all input data are assumed to be integers. Moreover, we demand $c_i \leq C$ for all $i \in I$, because the problem would become infeasible otherwise. Then, the TBBP-FU requires to find a job-to-server assignment minimizing the weighted sum of the number of servers required to accommodate all jobs and the number of fire-ups caused by the specific item sequence on every server. Consequently, an *instance* of the problem under consideration can be completely described by the tuple $E = (n, C, \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{e}, \gamma)$ where $\boldsymbol{c}$, $\boldsymbol{s}$, and $\boldsymbol{e}$ are n-dimensional vectors collecting the input-data (size, starting time, terminating time) of the items. Typically, we refer to the set of time instants by $T := \bigcup_{i\in I}\{s_i, e_i\}$, and address the set of starting times by $T_S = \bigcup_{i \in I}\{s_i\}$. Then, a subset of jobs can feasibly be assigned to a single server, if and only if for any $t \in T$ the capacity of this server is respected.

Minimizing the number of fire-ups and servers required is a very new aspect in the context of energy-efficient job-to-server scheduling introduced in [4]. In that article, this holistic goal has been addressed by two ILP formulations each of which being based on assignment variables and the well-known Kantorovich-type structure [39]. In what follows, we briefly present these two approaches and discuss their relationships as well as reduction methods already applied in the literature.

Let $K := \{1, \ldots, n\}$ denote the set of servers. Then, the first model introduced in [4] requires four different types of variables:

- The decision whether server $k \in K$ is used at all will be reflected by the binary variable $z_k \in \{0, 1\}$.
- For any $(i, k) \in I \times K$ we define a classical assignment variable $x_{ik} \in \{0, 1\}$ stating whether job $i \in I$ is performed on server $k \in K$ ($x_{ik} = 1$) or not ($x_{ik} = 0$).
- To display the activity 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 continuous variables $w_{tk} \geq 0$ with $k \in K$ and $t \in T$ contain information about the fact whether server k has been switched on at time t or not. (As we will see in a moment, these variables are decreased to either 0 or 1 in any optimal solution of the integer model, so that the fire-ups can be easily reconstructed.)

Moreover, we compute a parameter $a_{it} \in \{0,1\}$ for any $i \in I$ and $t \in T$ telling us whether job i is active at time t or not. More formally, we have $a_{it} = 1$ if and only if $t \in [s_i, e_i)$ holds. Then, according to [4], we obtain the

Assignment Model Type 1 (Model 1)

$$z^{(1)} = \gamma \cdot \sum_{k \in K} \sum_{t \in T} w_{tk} + \sum_{k \in K} z_k \to \min$$

$$\begin{array}{llrr}
\text{s.t.} & y_{tk} \leq \sum_{i \in I} c_i \cdot a_{it} \cdot x_{ik} \leq y_{tk} \cdot C, & k \in K, t \in T, & (1) \\
& \sum_{k \in K} x_{ik} = 1, & i \in I, & (2) \\
& x_{ik} \leq y_{s_i,k}, & i \in I, k \in K, & (3) \\
& y_{tk} \leq z_k, & k \in K, t \in T, & (4) \\
& y_{tk} - y_{t-1,k} \leq w_{tk}, & k \in K, t \in T_S, & (5) \\
& x_{ik} \in \{0,1\}, & i \in I, k \in K, & (6) \\
& y_{tk} \in \{0,1\}, & k \in K, t \in T, & (7) \\
& w_{tk} \geq 0, & k \in K, t \in T, & (8) \\
& z_k \in \{0,1\}, & k \in K. & (9)
\end{array}$$

The objective function minimizes a weighted sum of the number of all fire-ups (first term) and all used servers (second term). Moreover, the following conditions are required:

- The inequality on the right side in (1) requests that the capacity condition is met on a server $k \in K$ activated at time $t \in T$ (i.e., for $y_{tk} = 1$). However, this still allows to leave an actually empty server activated in order to save an additional fire-up later. The inequality on the left side, therefore, requires that at least one job must be allocated to a server activated at time $t \in T$ or (in the opposite direction) that an empty server must definitely be switched off ($y_{tk} = 0$).

- Constraints (2) guarantee that each job will be executed on exactly one server.

- Conditions (3) couple the x- and y-variables in the following way: When job i is running on server k, this specific server must be active especially at the start time $t = s_i \in T_S$ of job $i \in I$ (i.e., we have $y_{s_i,k} = 1$).

- Constraints (4) couple the y- and z-variables. If server k is active at some instant of time $t \in T$, it must also be counted as a used server in the objective function (i.e., $z_k = 1$).

- Restrictions (5) couple the y- and w-variables. A fire-up for server $k \in K$ must be registered at time $t \in T$ if this server was inactive at the previous[2] instant of time (i.e., $y_{t-1,k} = 0$) and has been switched on at time t (i.e., $y_{tk} = 1$). For any optimal solution, only in this case w_{tk} is assigned the value 1 (in all other cases we will have the value 0), since the objective function is minimized.

Remark 1. *In the unit commitment problem (UCP), the operation mode of thermal units has to be optimized, see [59] for a comprehensive survey, under constraints that can be seen as a generalization of (5). More precisely, for the UCP, whenever a unit was switched off it has to remain inactive for a given minimum amount of time (called the minimum down-time), see*

[2]For simplicity, here the predecessor of t in the set T is denoted by $t-1$, even if the actual difference between both instants of time is greater than one. Moreover, for the very first element $t := t_{\min}$ in the (chronologically ordered) set T, we will use $y_{t-1,k} := 0$ for all $k \in K$ as an initializing boundary condition.

[5, 6, 32]. Likewise, a unit that was activated at least has to stay in this mode for a period specified by a given minimum up-time. Moreover, one has to keep in mind that many other conditions have to be met (for the UCP) and these typically lead to nonlinear optimization problems. One such additional condition is, for example, that after switching on the unit, the entire power of a unit is not immediately available or, in the opposite direction, a unit cannot be switched from full load to idle operation (referred to as a ramp-up or ramp-down constraint, respectively). While all these additional conditions may also be important for job-to-server scheduling, we will only consider the original formulation of the problem from [4] here, but would like to keep an eye on the aforementioned extensions if the practical framework of the problem requires it in the future.

Altogether, with $|I| = n$ and $|K| = n$ (for the case with $\gamma > 1/n$ we consider), we obtain

$$\begin{aligned} n_{var} &= |I| \cdot |K| + 2|K| \cdot |T| + |K| = n^2 + 2n|T| + n = n^2 + n \cdot (2|T| + 1) \leq 5n^2 + n, \\ n_{con} &= |K| \cdot |T| + |I| + |I| \cdot |K| + |K| \cdot |T_S| = n|T| + n + n^2 + n|T_S| \\ &= n^2 + n \cdot (|T| + |T_S| + 1) \leq 4n^2 + n \end{aligned}$$

for the numbers of variables and constraints (thanks to $|T| \leq 2n$ and $|T_S| \leq n$).

The motivation behind the second approach proposed in [4] is to invest a little more effort in preprocessing in order to (mostly) have fewer variables and constraints in the ILP formulation itself. In particular, the temporal aspect of the optimization problem will then be addressed in a less direct manner in the model. To this end, we sort all jobs according to non-decreasing start times s_i (where ties are broken in an arbitrary way) and define the following two sets

$$\begin{aligned} \delta_i &:= \{j < i \mid s_i < e_j\}, \\ \delta_i^+ &:= \{j < i \mid s_i \leq e_j\}, \end{aligned}$$

for each $i \in I$. For given $i \in I$, the first set collects all jobs $j \in I$, $j \neq i$, which are active at the start time of i, whereas the second set gathers all jobs that are still active at the start time of i or that have just stopped at that moment. Compared to Model 1, we can now get rid of the y-variables and only require the w-variables at the possible starting times $t \in T_S$ (instead of the whole set T). More precisely, we obtain the

Assignment Model Type 2 (Model 2)

$$z^{(2)} = \gamma \cdot \sum_{k \in K} \sum_{t \in T_S} w_{tk} + \sum_{k \in K} z_k \to \min$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{j \in \delta_i} c_j x_{jk} + c_i x_{ik} \leq C \cdot z_k, && i \in I, k \in K && (10) \\ & \sum_{k \in K} x_{ik} = 1, && i \in I, && (11) \\ & x_{ik} \leq z_k, && i \in I, k \in K, && (12) \\ & \sum_{j \in \delta_i^+} x_{jk} - x_{ik} + w_{s_i,k} \geq 0, && i \in I, k \in K, && (13) \\ & x_{ik} \in \{0, 1\}, && i \in I, k \in K, && (14) \\ & w_{tk} \geq 0, && t \in T_S, k \in K, && (15) \\ & z_k \in \{0, 1\}, && k \in K. && (16) \end{aligned}$$

Again, the objective function displays the weighted sum over all fire-ups (first term) and all servers in use (second term). The following applies to the conditions:

- Constraints (10) require the jobs not to overload the servers' capacities. If server $k \in K$ is used (i.e., $z_k = 1$), then the jobs running parallel to $i \in I$ (that is, the set δ_i) together with i must never exceed the capacity of the server. (Since this is demanded for any $i \in I$, the time dimension is indirectly covered here.).

- Conditions (11) make sure that each job is assigned exactly once.
- Restrictions (12) couple the x- and z-variables in a sense that, whenever a job $i \in I$ is assigned to some server $k \in K$, this server has to be counted in the objective function. Note that this condition is redundant for the integer program, but it improves the LP bound[3].
- Constraints (13) are responsible for registering a fire-up ($w_{s_i,k} = 1$) only in those cases where server k did not possess any load precisely before the starting time of job i (i.e., we have $x_{jk} = 0$ for all $j \in \delta_i^+$) and, in addition, $x_{ik} = 1$ is true (i.e., the job i then causes the fire-up).

Due to

$$n_{var} = |I| \cdot |K| + |T_S| \cdot |K| + |K| = n^2 + n \cdot |T_S| + n = n^2 + n \cdot (|T_S| + 1) \leq 2n^2 + n,$$

Model 2 always has fewer variables than Model 1. As far as the number of constraints is concerned, we first count them

$$n_{con} = |I| \cdot |K| + |I| + |I| \cdot |K| + |K| \cdot |I| = 3n^2 + n,$$

and then find that a typical instance generally leads to smaller values for Model 2 due to $|T| \approx 2n$ and $|T_S| \approx n$. However, on the other hand, it can be shown that Model 2 provides worse LP bounds, see [4, Prop. 4.1], so that it is usually not considered in the computational experiments.

Remark 2. *Although [4, Prop. 4.1] also states an example, where the LP values $z_{LP}^{(1),\star}$ and $z_{LP}^{(2),\star}$ of both models are different (more precisely, we had $z_{LP}^{(1),\star} = 8/3$ and $z_{LP}^{(2),\star} = 7/3$), the rounded-up integer values obtained from these bounds are the same. To overcome this slight weakness, here we present a better example showing that even the integer-valued LP bounds can be different. Let us consider the instance E described by $n = 4$, $C = 3$, and the data contained in Tab. 1.*

i	1	2	3	4
s_i	1	1	1	3
e_i	2	2	2	4
c_i	2	2	2	2

Table 1: An exemplary instance leading to different integer-valued LP bounds for Model 1 and Model 2

Obviously, the set T_S of starting times is given by $T_S = \{1, 3\}$. Then, we obtain the optimal value $z^{(1),\star} = 5$ for Model 1 using two servers (i.e., $z_1 = z_2 = 1$) and three fire-ups, caused by the decisions displayed in Tab. 2.

x_{ik}	1	2
1	1/2	1/2
2	0	1
3	1	0
4	0	1

w_{tk}	1	2
1	1	1
3	0	1

Table 2: An optimal solution of Model 1 (represented by the most important variables only)

Contrary to this, we can find feasible points for Model 2 with two servers and only two fire-ups (meaning that $z^{(2),\star} \leq 4$ holds), as listed in Tab. 3.

[3]Having $x_{ik} = 1$ for some $(i, k) \in I \times K$ in the LP relaxation now leads to $z_k = 1$. Without Conditions (12), the choice $x_{ik} = 1$ normally just implies $z_k \geq c_i/C$ thanks to (10).

x_{ik}	1	2
1	3/14	11/14
2	6/14	8/14
3	12/14	2/14
4	11/14	3/14

w_{tk}	1	2
1	3/14	11/14
3	11/14	3/14

Table 3: An optimal solution of Model 2 (represented by the most important variables only)

Hence, the second model will lead to a worse (rounded-up) LP bound for the discrete optimal value.

The reason for the different behavior of the LP bounds is mainly related to the different ways the variables are coupled within the two models. In the first model, a good server utilization (in terms of capacity) already leads to relatively large y-values (due to Conditions (1)), which in turn directly influence the w-variables. Thanks to Constraints (5), particularly the first fire-up of a server is then connected to a relatively large w-variable. Since, in the second model, only the x-variables influence the w-variables (and not the capacity utilization of the servers), the latter can often attain much smaller values as to be clearly seen in our previous exemplary instance.

In the following we would like to present some possible improvements for both formulations, the aim of which is to harmonize the LP bounds so that the lower complexity of the second model might lead to certain advantages in the numerical experiments after all. Afterwards, a new ILP approach will be introduced, and the applicability of reduction techniques will be studied also with respect to this formulation.

3. Reduction Methods

Obviously, the two ILP formulations are somewhat based on assignment models of Kantorovich-type for the ordinary bin packing problem. It is a well known fact that such approaches are equipped with two main drawbacks:

- Typically, the LP bound is rather poor, meaning that it can be arbitrarily far away from the true optimal value (of the ILP).
- The feasible region includes a large number of symmetric solutions, which leads to significant additional efforts for branch-and-bound based solution techniques.

To tackle these challenging aspects, two ideas were already introduced in [4]:

(Lit-A) A lower bound $h \in \mathbb{N}$ for the number of active servers can be implemented to raise the LP value. More precisely, the authors of [4] suggest to use

$$h_0 = \left\lceil \max_{t \in T} \left\{ \frac{\sum_{i \in I} a_{it} c_i}{C} \right\} \right\rceil, \tag{17}$$

which could be termed as the *material bound*, and add the constraint $z_1 + \ldots + z_{h_0} = h_0$ to the ILP model. Note that this lower bound also appears in [26, Property 1] for the ordinary TBPP, where it is derived from a decomposition strategy with respect to the temporal dimension. Moreover, it is shown that this bound can be obtained in $\mathcal{O}(n \cdot \log(n))$, see [26, Property 2].

(Lit-B) The servers can be sorted with respect to their "activity level" by using $z_k \geq z_{k+1}$ for $k = 1, \ldots, |K| - 1$. This helps to reduce the number of possible permutations we can apply to feasible points without changing their objective value.

Note that these techniques (and also the general modeling) can be made a bit more efficient.

- Basically, it would be sufficient to only consider starting times $t \in T_S$ to find the maximum value in (17), since the total load on all servers definitely decreases when dealing with an end time $t \in T \setminus T_S$. Apart from this, however, the use of better lower bounds is recommended, in general. Research has shown that the material bound h_0 is dominated by a lower bound $h := h_{CG}$ obtained from solving the LP relaxation of a pattern-based model for the traditional TBPP by means of *column generation*, see [26, Property 4]. Since this typically better value can also be calculated in a very short time, as was reported in [26], we strictly follow the implementation details specified in [26, Subsect. 6.1] here and apply a general-purpose solver[4] to the arising subproblems (that is, a TKP, see [26, Eq. (17)]).
- On the other hand, after having installed the lower bound h, we only need to demand $z_k \geq z_{k+1}$ for $k = h+1, \ldots, |K|-1$ because all other inequalities of this type are automatically satisfied.
- In addition, we also assume the w-variables to be binary since this has lead to slight numerical advantages in our internal precalculations. On the one hand, we attribute this observation to the reduction of the search space, but more importantly, binary w-variables can be chosen for branching ($w_{tk} = 0$ vs. $w_{tk} = 1$), where at least in the latter case the objective value (i.e., the bounds) of the subproblems are likely to increase.

Altogether, we will refer to this refined version of the state-of-the-art version of Model 1 by the abbreviation (Lit).

To further improve the first model, we propose the following additional steps:

(R1) **Reducing variables and symmetries:** In addition to the efforts made in (Lit-B), a quite efficient way to limit the number of symmetric solutions is given by an implicit renumbering of the active servers. More precisely, we can demand that job $i = 1$ is assigned to server $k = 1$. Then, job $i = 2$ can either be processed on the same ($k = 1$) or on a new server ($k = 2$), and so on. In general, any job i can only be assigned to the servers $k \leq i$, so that we can considerably reduce the set of x-variables from $(i,k) \in I \times K$ to $(i,k) \in \Delta$ with $\Delta := \{(i,k) \in I \times K \,|\, k \leq i\}$. By doing so, roughly half of the x-variables (i.e., a quadratic number with respect to n) can be removed from the ILP formulation.

Moreover, we mention that the "triangular structure" of the x-variables also influences some of the other variable types. The introduction of set Δ entails that it is no longer possible to arrange each job on any arbitrary server. Consequently, the times in general and, particularly, the starting times that have to be modeled on each server are now different, which is addressed by the server-dependent sets

$$\begin{aligned} T(k) &= \{t \in T \mid \exists i \in I : t \in \{s_i, e_i\}, (i,k) \in \Delta\} = \bigcup_{i \geq k} \{s_i, e_i\}, \\ T_S(k) &= \{t \in T_S \mid \exists i \in I : t = s_i, (i,k) \in \Delta\} = \bigcup_{i \geq k} \{s_i\}. \end{aligned}$$

Based on these definitions, we can restrict the sets of variables to w_{tk} with $k \in K$, $t \in T_S(k)$, and y_{tk} with $k \in K$, $t \in T(k)$. But then, we have to pay attention that – depending on the specific server k – our simplified notation $t-1$ can refer to different instants of time, because we now have to take the predecessor of t in $T(k)$.

(R2) **Valid inequalities:** To improve the LP bound feasible cuts can be added to the integer problem. These constraints do not affect the optimal value of the integer problem, but

[4] As will be explained later in Sect. 5 when reporting about the computational experiments, we use Gurobi for this.

they can lead to better values if continuous variables are considered. More precisely, the following inequalities[5] can be applied:

(a) Any activated server has to produce at least one fire-up meaning that we can demand

$$z_k \leq \sum_{t \in T_S(k)} w_{tk} \tag{18}$$

for any $k \in K$.

(b) Whenever a fire-up is noticed on server $k \in K$, then it has to be caused by one specific job that has been assigned to this server. Hence, we can demand

$$\sum_{t \in T_S(k)} w_{tk} \leq \sum_{i \in I:(i,k) \in \Delta} x_{ik} \tag{19}$$

for any $k \in K$.

(c) An activated server has to accommodate at least one job, meaning that

$$z_k \leq \sum_{i \in I:(i,k) \in \Delta} x_{ik} \tag{20}$$

has to hold for any $k \in K$. (Note that this valid inequality is implied by the previous two classes.)

(d) A server that is counted in the objective function has to be activated at least once during the time horizon considered. So, we can add the condition

$$z_k \leq \sum_{t \in T(k)} y_{tk} \tag{21}$$

for all $k \in K$.

(R3) **Lifting:** The term *lifting* describes a traditional preprocessing concept in cutting and packing [11, 17], see [26, Sect. 4] for two ideas related to the TBPP. Roughly speaking, the aim of this procedure is to transform a given instance E into another instance $\widetilde{E}$ having the same set of integer feasible solutions, but possessing computational advantages (e.g., in terms of the LP relaxation). In our setting, we apply the first strategy from [26] to increase the item sizes c_i, $i \in I$. To be more precise, we first define the sets

$$A(i) := \{j \in I \setminus \{i\} \mid [s_i, e_i) \cap [s_j, e_j) \neq \emptyset\}$$

for $i \in I$, denoting all jobs that overlap with i in a temporal sense. Based on this, we compute

$$\varepsilon(i) := \max \left\{ \sum_{p \in A(i)} c_p x_p \;\middle|\; \sum_{p \in A(i)} c_p x_p \leq C - c_i,\, x_p \in \{0,1\} \text{ for all } p \in A(i) \right\},$$

i.e., the maximum capacity required (on a single server) by a feasible assignment that contains job i. Whenever $\varepsilon(i) < C - c_i$ is observed, the parameter c_i can be enlarged to $\widetilde{c}_i := c_i + (C - c_i - \varepsilon(i)) = C - \varepsilon(i)$. Basically, the effects of this technique are twofold:

[5]In our final computations, we will mostly apply all possible reductions together to obtain the most promising ILP formulation. Due to this reason, here we do not formulate the valid inequalities for the original version (Lit) of Model 1. Instead, we already include the reductions described in the previous point (R1).

(a) Rising the parameters c_i can lead to an increased material bound h_0, so that more variables can be fixed prior to the optimization. However, since we directly use the better lower bound h (obtained by column generation), this fact is not significant for our calculations.

(b) More importantly, increasing the parameters c_i means that some of the coefficients in Constraints (1) become larger, so that (for the LP relaxation) the conditions become more restrictive and may lead to a better LP value.

In a first step, we highlight the important facts that

- any of the proposed individual reductions can already be beneficial to rise the LP value,
- combining all the reductions can additionally boost the performance.

Example 1. *Let us consider an instance E defined by $n = 4$, $C = 3$ and the data contained in Tab. 4. For the sake of a better readability, we also provide a graphical visualization of this instance in Fig. 4.*

i	1	2	3	4
s_i	1	1	3	3
e_i	3	2	4	4
c_i	2	3	1	3

Table 4: An exemplary instance to show the benefits of the various reduction methods

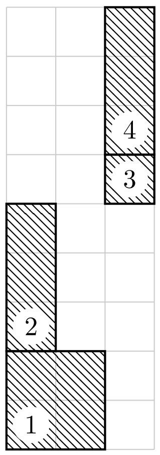


Figure 4: Illustration of the instance E

Note that the optimal value of this instance is given by $z^{(1),\star} = 5$ (caused by two servers and three fire-ups). In Tab. 5, we just collect the optimal values of the LP relaxation for different steps of reduction. The corresponding optimal solutions can be found in AppendixA.

formulation	$z_{LP}^{(1),\star}$
(Lit)	11/3
(Lit) + (R1)	4
(Lit) + (R2)	4
(Lit) + (R3)	4
(Lit) + (R1)-(R3)	5

Table 5: Optimal LP value for the instance from Tab. 4

As to be clearly seen, any single reduction leads to an improved LP value, while gathering all improvements already gives an integer optimal solution.

Remark 3. *In our internal precalculations, we noticed that only Conditions (18) lead to any benefits in terms of the LP value. Although this may seem surprising at the first glance, the reasons for this behavior are quite well understandable. To be more precise, the objective function of Model 1 aims at keeping the values of the z- and the w-variables small. Consequently, valid inequalities can rise the LP value particularly in those cases, where lower bounds for any of these variables are imposed. The only class of cuts proposed in (R2) contributing to this goal is given by (18). Hence, we will only use this set of inequalities in the final numerical experiments, while the others, i.e., categories (19)-(21), are intended to remain in the list for the sake of completeness.*

In terms of the second ILP formulation (Model 2), no reductions have previously been presented in the literature. However, all the methods introduced for Model 1 can also be applied to Model 2 in a more or less straightforward way. To this end, here we only briefly collect the various steps of improvements:

(R0) **Off-the-shelf reductions from Model 1:** The first step contains three techniques that are a direct consequence of the reductions proposed for the first ILP formulation in [4]. To be more precise, here we also make use of

- the lower bound h to fix the value of some z-variables,
- the sorting $z_k \geq z_{k+1}$ for $k = h+1, \ldots, |K|-1$ to partly avoid symmetries,
- binary w-variables to reduce the search space.

(R1) **Reducing variables and symmetries:** It is again sufficient to only consider the combinations $(i,k) \in \Delta$ which also means that the index set of the w-variables can be changed to $k \in K$, $t \in T_S(k)$. Note that, due to the absence of the y-variables, this does not entail further reductions.

(R2) **Valid inequalities:** Also the second ILP formulation allows for several classes of valid inequalities. However, as observed earlier, those cuts imposing lower bounds on the variables appearing in the objective function are particularly promising. Due to this reason, here we only mention a single set of additional constraints, namely Conditions (18) presented above, but not without pointing out that further valid inequalities can easily be obtained, while they did not lead to any gains in our internal test.

(R3) **Lifting:** This technique can be executed in precisely the same manner as before.

Moreover, the second ILP formulation allows for two additional modifications which will be termed as (R4) and (R5):

(R4) **Strengthening Constraints (13):** Let us consider the definition of the sets δ_i^+ that are used in Conditions (13), and assume that there are two (or more) jobs $p, q \in I$ having the same starting time (see Tab. 1 for an example). Then, depending on the sorting, we either have $p, q \in \delta_p^+$ or $p, q \in \delta_q^+$, where the latter is chosen here without loss of generality. Hence, for some server $k \in K$, the constraints of type (13) belonging to $i = p$ and $i = q$ would be as follows

$$\begin{aligned} i = p \quad &: \quad w_{tk} \geq x_{pk} - \sum_{j \in \delta_p^+} x_{jk}, \\ i = q \quad &: \quad w_{tk} \geq x_{qk} - \sum_{j \in \delta_q^+} x_{jk} = x_{qk} - \sum_{j \in \delta_p^+} x_{jk} - x_{pk}, \end{aligned}$$

where $t = s_p = s_q$ describes the joint starting time of both jobs. In particular, the condition for $i = q$ also depends on the item p, but this item is not relevant to check whether q causes a fire-up at time t. Hence, we can get rid of x_{pk} in the corresponding condition of type (13). Observe that this may lead to a stronger inequality (for the LP relaxation) and to fewer nonzero elements in the constraint matrices. Altogether, we recommend to modify the previous definition of δ_i^+ (from the literature) as follows

$$\delta_i^+ := \{j < i \,|\, s_i \leq e_j, s_i \neq s_j\} \tag{22}$$

to exclude items having the same starting time from the summation appearing in Conditions (13).

(R5) **Temporal dominance:** If we can find two directly successive starting times $t_1 < t_2 \in T_S$ (with $t_2 \notin \bigcup_{i \in I}\{e_i\}$ not representing an end time) in the chronologically ordered set T, then we know that all jobs, that are active at t_1, are still active at t_2. Under these conditions, we say that t_2 *dominates* t_1, because the capacity constraint (10) for $t = t_2$ already implies that of $t = t_1$ (as no job is terminating in the interval $[t_1, t_2]$). Let us define

$$T_S^{nd} := \left\{ t \in T_S \,\middle|\, succ_T(t) \in \bigcup_{i \in I}\{e_i\} \right\},$$

with $succ_T(t)$ indicating the successor of t in T, then $T_S^{nd} \subseteq T_S$ represents the set of *non-dominated* starting times, see also [26, Sect. 3]. Hence, considering Conditions (10) for $i \in I$ with $s_i \in T_S^{nd}$ is sufficient.

Altogether, the structure of the second formulation allows for additional reductions which may possibly counterbalance the numerical drawbacks (previously reported in [4]) of this approach. This issue will be thoroughly investigated in the numerical part of this paper.

4. A New Compact ILP Formulation

As a second main contribution of this article, we would like to present a new ILP formulation (offering further reduction potentials) that requires to rephrase the verbal interpretation of the assignment variables previously used. To this end, instead of addressing a job-to-server scheduling, we now focus on a job-to-job correspondence by defining $x_{ik} \in \{0,1\}$ with $x_{ik} = 1$ if and only if job i is executed on a server that was initialized by job k. By *initialized* we mean that job k was the first job (in a temporal sense) assigned to the respective server, so that job k definitely contributed to a fire-up[6]. By this new interpretation, we indirectly attached a temporal relationship to the assignment variables, so that they are obviously only required for index pairs $(i, k) \in \Delta$. For the sake of simplicity, x_{kk}, $k \in I$, should always be understood as the decision whether job k initializes a server or not.

Remark 4. *Note that, originally, Δ has been introduced as a subset of $I \times K$. Although now collecting elements of $I \times I$, we keep the same symbol since the essential part of the definition (that is, $k \leq i$) does not change. Moreover, as we cannot reduce the number of possible servers by heuristic approaches in advance, we always have $I = K$ in our scenarios, so that no misunderstanding will be caused by this notation.*

[6]If there are several jobs with the same starting time on the same server and if this starting time is connected to a fire-up on the considered server, then the lowest-indexed job will be chosen (by the structure of the model) to cause the fire-up.

Similarly, also the number and the interpretation of the w-variables changes. More precisely, we define $w_i \in \{0,1\}$, $i \in I$, with $w_i = 1$ indicating that job i caused a fire-up on whatever server. As a consequence of this new point of view, we formulate the

Assignment Model Type 3 (Model 3)

$$z^{(3)} = \gamma \cdot \sum_{i \in I} w_i + \sum_{k \in I} x_{kk} \to \min$$

$$\text{s.t.} \quad \sum_{j \in \delta_i} c_j x_{jk} + c_i x_{ik} \leq C \cdot x_{kk}, \qquad (i,k) \in \Delta \tag{23}$$

$$\sum_{k \in I : (i,k) \in \Delta} x_{ik} = 1, \qquad i \in I, \tag{24}$$

$$x_{ik} \leq x_{kk}, \qquad (i,k) \in \Delta, i \neq k, \tag{25}$$

$$\sum_{j \in \delta_i^+} x_{jk} - x_{ik} + w_i \geq 0, \qquad (i,k) \in \Delta, \tag{26}$$

$$x_{ik} \in \{0,1\}, \qquad (i,k) \in \Delta, \tag{27}$$

$$w_i \in \{0,1\}, \qquad i \in I. \tag{28}$$

In comparison to Model 1 and Model 2, it can immediately be noticed that

- there is no symmetry (even in the unimproved setting),
- only a linear number of w-variables is required,
- we can get rid of the z-variables,

so that we end up with

$$\begin{aligned} n_{var} &= |\Delta| + |I| = \frac{n(n+1)}{2} + n, \\ n_{con} &= |\Delta| + |I| + (|\Delta| - |I|) + |\Delta| = 3|\Delta| = \frac{3}{2} \cdot n(n+1), \end{aligned}$$

i.e., a (mostly) considerably less complex ILP formulation. On top of that, the applicability of further reductions will be discussed in the following list. Since Model 3 already starts with a certain "lead" (in terms of complexity), not all the previous reductions can still be applied. We will see that some have become obsolete due to the new variable interpretation, while others require significant changes. In the latter case, we use the symbol $\star$ to distinguish this reduction (which is named in the same way as before, but in detail is different) from the previous terms.

(R0) **Off-the-shelf reductions from Model 1:** Here we can only make use of the lower bound h by demanding

$$h \leq \sum_{k \in I} x_{kk}.$$

Observe that the sorting of the z-variables is no longer possible, while the w-variables were already introduced as binary decisions.

(R1*) **Reducing variables and symmetries:** The original reductions (related to the introduction of the set Δ) are now automatically included already by definition. However, due to the new job-to-job assignment, further improvements are possible because a pair $(i,k) \in \Delta$ with $i \neq k$ can only be executed on the same server if both jobs (i and k) do not overlap or if they do not exceed the capacity C of the server. More formally, considering

$$\Delta_{red} := \{(i,k) \in \Delta \mid \langle i \neq k, [s_k, e_k) \cap [s_i, e_i) \neq \emptyset \Longrightarrow c_i + c_k \leq C \rangle\} \tag{29}$$

is sufficient. All other index pairs from Δ would automatically lead to a zero variable, so they do not have to be considered. This reduction is particularly promising if there are lots of (temporal) interactions between the jobs and if the corresponding item sizes are relatively large.

(R2*) **Valid inequalities:** The only useful valid inequality, previously termed as Conditions (18), could now be written as

$$x_{kk} \leq w_k \tag{30}$$

for any $k \in I$, because we know that a server which has been initialized by job k has to count a fire-up precisely caused by that item. Observe that the right hand side of (30) only contains one specific term from the sum that previously appeared in (18), thus describing a stronger inequality. Moreover, this much more direct connection between the x- and the w-variables could also contribute to easier subproblems in the branching trees. While up to now (in Model 1 and Model 2), we had to know that $w_{tk} = 0$ holds for all $t \in T_S(k)$ to conclude $z_k = 0$ (or in the reverse direction, $z_k = 1$ did not specify the value of a single w-variable), now already $w_k = 0$ implies $x_{kk} = 0$ (or, alternatively, $x_{kk} = 1$ entails $w_k = 1$). Consequently, after having chosen a variable to branch it now becomes more likely that additional variables directly obtain integer values.

(R3) **Lifting:** This technique can be executed in precisely the same manner as before.

(R4) **Strengthening Constraints (26):** Here it is not possible to exclude the items having the same starting time from the definition of δ_i^+. In order to better understand this, we take the following case as an example: Let $i < j \in I$ with $s_i = s_j$ be given and assume that both jobs have been attached to some initializing job $k \in I$, meaning that we have $x_{ik} = x_{jk} = 1$. Then, we would obtain $\delta_i^+ = \delta_j^+$ for the sets defined according to (22). If these two sets are empty (or, more generally, if the sum of the corresponding x-variables is zero), then Constraints (26) would lead to $w_i = w_j = 1$, i.e., two fire-ups would be counted which is not correct. Hence, we cannot use this type of reduction here.

(R5*) **Temporal dominance:** Since we have completely deleted the temporal aspect from the variable indices, referring to non-dominated starting times is not reasonable for the current model. Instead, the following implication can easily be verified

$$i < j \in I,\, s_i = s_j \Longrightarrow \{i\} \cup \delta_i \subseteq \{j\} \cup \delta_j, \tag{31}$$

so that (for any suitably chosen $k \in I$) the left hand side of (23) for $(i,k) \in \Delta_{red}$ is completely contained in the left hand side of (23) for $(j,k) \in \Delta_{red}$, while the right hand side does not change. Consequently, the capacity constraint corresponding to (j,k) *dominates* the condition associated with (i,k). Roughly spoken, if there are two (or more) jobs having the same starting time, and both of them are allowed to be executed together with a fixed initializing job $k \in I$, then the capacity constraint only has to be satisfied for the highest-indexed job (with respect to the chronologically sorted list that was required to define the sets δ_i and δ_i^+). More formally, we state: Let $k \in I$ be fixed. If there are two jobs $i, j \in I$ with $k < i < j$, $(i,k), (j,k) \in \Delta_{red}$ and $s_i = s_j$, then (j,k) dominates (i,k). To be able to represent this observation as easily as possible in mathematical language, we use the set $\Delta_{red}^{nd} \subseteq \Delta_{red}$ to refer to the *non-dominated index pairs.*

To account for the fact that this compact model is completely new, we explicitly formulate it in its maximally reduced shape as the

(Reduced) Assignment Model Type 3 (Model 3)

$$z^{(3)} = \gamma \cdot \sum_{i \in I} w_i + \sum_{k \in I} x_{kk} \to \min$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{j \in \delta_i : (j,k) \in \Delta_{red}} c_j x_{jk} + c_i x_{ik} \le C \cdot x_{kk}, && (i,k) \in \Delta_{red}^{nd}, i \neq k, && (32)\\
& \sum_{k \in I : (i,k) \in \Delta_{red}} x_{ik} = 1, && i \in I, && (33)\\
& x_{ik} \le x_{kk}, && (i,k) \in \Delta_{red}, i \neq k, && (34)\\
& \sum_{j \in \delta_i^+} x_{jk} - x_{ik} + w_i \ge 0, && (i,k) \in \Delta_{red}, && (35)\\
& x_{ik} \in \{0,1\}, && (i,k) \in \Delta_{red}, && (36)\\
& w_i \in \{0,1\}, && i \in I. && (37)
\end{aligned}
$$

Note that we exclude the possibility $i = k$ in Conditions (32), because this constellation always leads to trivial inequalities of type $c_i x_{kk} \le C x_{kk}$. Indeed, for $i = k$, conditions $j \in \delta_k^+$ and $(j,k) \in \Delta_{red}$ become incompatible, as the first requires $j < k$ while the latter demands $j \ge k$. Consequently, the index set of the sum appearing in (32) is empty for precisely those cases, and we do not need to generate the respective constraint. For the sake of completeness, we also highlight that this observation is responsible for demanding $k < i$ in the definition of non-dominated index pairs, because the diagonal elements (k,k) (for appropriately chosen $k \in K$) can be ignored due to the aforementioned explanation.

Whenever there are different modeling approaches for one and the same optimization problem, an important question deals with the strength of the (integer-valued) bounds obtained from the corresponding LP relaxations. It is well known that the tightness of a relaxation is a crucial factor in the size of branch-and-bound trees, because it significantly influences the efforts to solve the ILP. Hence, before concluding the current section, we present several very small exemplary instances to show that there are no dominance relations between the three LP relaxations (obtained from the three reduced assignment models). For the sake of simplicity, we partly refer to the models by their respective abbreviations M1, M2, and M3. These symbols will also appear later in the computational part, see Sect. 5.

Theorem 5. *There are no dominance relations between the rounded-up (optimal) LP values of M1, M2, and M3.*

Proof. We proof this claim by three different examples:

- Let us consider an instance with $C = 2$ and $n = 3$ items characterized by $\boldsymbol{s} = (1,1,3)$, $\boldsymbol{e} = (2,4,4)$, and $\boldsymbol{c} = (1,2,1)$, see the leftmost picture in Fig. 5. Here, we obtain $\lceil z_{LP}^{(1),\star} \rceil = \lceil z_{LP}^{(2),\star} \rceil = 5$ and $\lceil z_{LP}^{(3),\star} \rceil = 4$.
- Let us consider an instance with $C = 2$ and $n = 4$ items characterized by $\boldsymbol{s} = (1,1,1,3)$, $\boldsymbol{e} = (2,4,2,4)$, and $\boldsymbol{c} = (1,1,2,2)$, see the middle picture in Fig. 5. Here, we obtain $\lceil z_{LP}^{(1),\star} \rceil = \lceil z_{LP}^{(3),\star} \rceil = 5$ and $\lceil z_{LP}^{(2),\star} \rceil = 4$.
- Let us consider an instance with $C = 3$ and $n = 3$ items characterized by $\boldsymbol{s} = (1,1,3)$, $\boldsymbol{e} = (2,4,4)$, and $\boldsymbol{c} = (2,3,3)$, see the rightmost picture in Fig. 5. Here, we obtain $\lceil z_{LP}^{(2),\star} \rceil = \lceil z_{LP}^{(3),\star} \rceil = 5$ and $\lceil z_{LP}^{(1),\star} \rceil = 4$.

The corresponding solutions of all LP relaxations can be found in AppendixB. □

We would like to point out in particular that – in contrast to the situation for the unimproved models (see Remark 2) – the dominance (in terms of the LP bound) between M1 and M2 no longer applies, so that our reductions eliminated an additional drawback of the second assignment model.

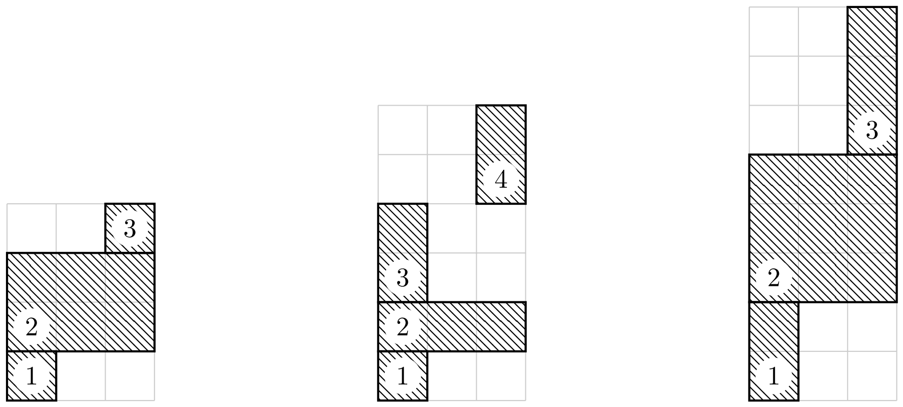


Figure 5: Illustrations of the instances used within the proof of Theorem 5.

5. Numerical Experiments

In this section, we collect the results of our computational experiments which have been conducted to compare the numerical properties and performances of the approaches presented before. To this end, we coded the three formulations in Python (version 3.7.7) and used its Gurobi (version 9.0.2) interface to solve the corresponding ILP models. All the experiments were run on an AMD A10-5800K processor with 16 GB RAM. Moreover, following the conventions of [4], a time limit of $t_{\max} = 30$ minutes was chosen if not stated otherwise.

5.1. Data Sets and Methodology

Since the TBPP is a relatively young field of research, there is not yet a comprehensive collection of associated benchmark instances. However, in the relevant literature, two major data sets have been introduced to study the behavior of exact approaches. Even if they were not specifically designed for the TBBP-FU, we will consider the following instance categories:

(A) In [4, Sect. 5], the authors presented 160 instances divided into 32 groups having 5 instances each. Any of these groups is determined by the bin capacity $C = 100$, a fixed number $n \in \{50, 100, 150, 200\}$ of items, and three additional indicators:

- **Time horizon:** It is assumed that the starting times s_i, $i \in I$, are uniformly distributed on $[0, \bar{s}] \cap \mathbb{Z}_+$, where $\bar{s} \in \{n, 1.2n\}$ either represents a rather dense ($\bar{s} = n$) or a more relaxed ($\bar{s} = 1.2n$) scenario with respect to possible temporal interactions of the items.
- **Duration:** The *item durations* $d_i := e_i - s_i$, $i \in I$, either correspond to a *short* (i.e., $d_i \in [10, 30] \cap \mathbb{Z}_+$) or *long* (i.e., $d_i \in [20, 60] \cap \mathbb{Z}_+$) scenario. For the sake of simplicity, these constellations will briefly be referred to as d_S (for 'short') and d_L (for 'large') in the upcoming tables and explanations.
- **Item sizes:** To also account for a certain variety with respect to the capacity dimension, instances can be equipped with a *low* (i.e., $c_i \in [25, 50] \cap \mathbb{Z}_+$) or a *high* (i.e., $c_i \in [25, 75] \cap \mathbb{Z}_+$) resource demand. In the following, we will refer to these scenarios by c_L (for 'low') and c_H (for 'high').

For the sake of completeness, we note that further instances (with much larger values of n) are described in [4], but these are solved exclusively heuristically, since addressing the exact solution is expected to be too difficult. Therefore we will not consider these additional instances here.

(B) The instances presented in [31, Sect. 7] use $C = 100$, uniformly distributed item sizes $c_i \in [10, 100]$, and they are based on a testbed originally introduced for the temporal knapsack problem in [14]. While the full details of this construction can be found in [31], here we only state that these instances particularly focus on the number of different instants of time $|T|$. More precisely, for any $|T| \in \{5, 10, 15, 20, 30, \ldots, 150\}$, a set of 100 instances[7] has been considered that are further divided into 10 classes (from I to X) of 10 instances each. The classes mainly differ in two ways:

   - For any time step, the number of active items is drawn from a uniform distribution on $[a_n, a_{\max}] \cap \mathbb{Z}_+$.
   - For any time step (except for the very first one), a percentage uniformly distributed in $[b_n, b_{\max}] \cap \mathbb{Z}_+$ is selected to indicate how many jobs are 'inherited' from the previous time step.

   Roughly speaking, the first decision manages the degree of parallelization while the second one indirectly influences the duration of the items. The full details regarding the precise choices of the parameters a_n, $a_{\max}$, b_n, and $b_{\max}$ in the ten different classes are given in [31, Tab. 1].

5.2. Complexity of the ILP Formulations for Category (A)

At first, we would like to focus on the complexity of the various approaches measured by the numbers of variables (n_{var}), constraints (n_{con}), and nonzero elements (n_{nz}) in the matrix formed by all restrictions. In a first experiment, we list the results for a subset[8] of the instances from Category (A) to obtain an overview on the effects of our reduction methods applied to the state-of-the-art models from the literature, see Tab. 6.

Remark 6. *As already explained in the theoretical part of our paper, the abbreviation (Lit) refers to a formulation that only contains the reductions previously presented in the literature, whereas (Red) includes all the additional improvements reported in Sect. 3. By way of example, for Model 1 (abbreviated by M1), the term (Red) is equivalent to (R1)-(R3), while for Model 2 (abbreviated by M2), it gathers all five steps of reduction, i.e., (R1)-(R5).*

Given the large number of instances attempted, we just provide average numbers instead of referring to every single instance individually. Remember that for Category (A) these averages are formed by the data obtained from five instances each. The main observations based on Tab. 6 are the following:

- Comparing the versions from the literature, we can obviously support the claim from [4] stating that Model 2 is considerably less complex than Model 1. This relation is also inherited by the reduced versions, so that the improved variant of Model 2 always possesses the most promising properties (i.e., the lowest numbers).
- For each of the three indicators (n_{var}, n_{con}, and n_{nz}), we see the large savings that resulted from our proposed reductions. More precisely, roughly 50% of the variables are saved for M2 (which is mainly caused by the triangular structure of the x-variables), while an even larger decrease can be observed for M1 (because, here, also the y-variables can be reduced significantly). In terms of constraints, we see that the reductions for Model 1 are typically slightly below 50%, whereas up to 58% fewer restrictions can be obtained for Model 2. This difference is mainly connected with the additional temporal dominance (R5) which can be formulated only for the second formulation.

[7]Note that we will not make use of all these 1700 instances, since they were designed for another optimization problem and sometimes only the approximate solution was addressed. Consequently, in the light of [31, Tab. 6] dealing with exact approaches for the TBPP, we mainly focus on those instances having moderate sets of time steps.

[8]The full details can be found in Tab. C.12 in AppendixC.

				n_{var}				n_{con}				n_{nz}			
				M1		M2		M1		M2		M1		M2	
n	$\bar{s}$	d_i	c_i	(Lit)	(Red)	(Lit)	(Red)	(Lit)	(Red)	(Lit)	(Red)	(Lit)	(Red)	(Lit)	(Red)
100	100	d_S	c_L	28.9	13.5	16.7	**8.4**	45.0	24.0	30.2	**13.8**	403.2	199.4	410.9	**168.1**
			c_H	29.1	13.5	16.3	**8.3**	45.0	24.1	30.2	**14.1**	408.2	203.4	412.7	**173.5**
		d_L	c_L	31.3	14.4	16.5	**8.4**	48.4	26.6	30.2	**13.7**	702.5	325.0	676.2	**243.0**
			c_H	30.9	14.3	16.3	**8.3**	47.6	26.4	30.2	**13.7**	702.3	329.7	686.8	**249.7**
	120	d_S	c_L	30.7	14.2	17.0	**8.6**	48.0	25.5	30.2	**13.9**	390.7	193.5	357.6	**147.1**
			c_H	30.8	14.2	17.0	**8.6**	48.1	25.6	30.2	**13.7**	387.0	194.4	352.8	**145.1**
		d_L	c_L	32.1	14.7	16.8	**8.6**	49.9	27.2	30.2	**13.8**	639.0	304.9	577.6	**219.9**
			c_H	32.8	15.0	17.0	**8.7**	51.2	27.7	30.2	**13.5**	678.7	320.3	597.6	**219.3**
200	200	d_S	c_L	112.6	52.5	66.1	**33.4**	174.9	91.0	120.4	**54.9**	1679.4	846.4	1741.6	**742.4**
			c_H	111.2	51.8	65.5	**33.1**	172.1	89.8	120.4	**56.2**	1608.9	814.7	1702.1	**747.6**
		d_L	c_L	118.3	54.8	66.1	**33.4**	183.4	97.8	120.4	**55.3**	3022.9	1472.2	3103.0	**1249.6**
			c_H	115.5	53.3	65.1	**32.9**	178.2	94.3	120.4	**54.9**	2873.5	1391.9	3049.8	**1227.0**
	240	d_S	c_L	120.8	55.1	67.7	**34.0**	188.8	97.6	120.4	**54.2**	1601.4	808.0	1489.1	**627.5**
			c_H	122.2	55.9	68.1	**34.4**	191.3	99.3	120.4	**54.2**	1628.3	826.7	1488.4	**634.3**
		d_L	c_L	125.0	56.9	68.0	**34.2**	195.4	102.6	120.4	**54.7**	2853.1	1391.4	2634.8	**1063.9**
			c_H	123.7	56.5	67.4	**34.1**	192.8	101.7	120.4	**55.0**	2782.8	1362.5	2609.2	**1060.9**

Table 6: Measuring the effects of our proposed reductions (for Model 1 and Model 2) for some representative sets of instances from Category (A). The numbers listed in the table indicate units of 10^3, and the best number per category is printed in boldface.

- By and large, we do not see any significant differences for the numbers of variables and constraints, when n and the specific model is fixed. As reported in Sect. 2 and Sect. 3, for a given model, these two numbers are mainly determined by the number n of jobs. Slight variations may, however, appear due to the specific input data of an instance, since they can influence the success of some of the reductions (like (R1) or (R5)).
- However, the number of nonzeros is typically much higher for d_L (compared to d_S), because there we have much more temporal interactions between the jobs so that, for example, the sets δ_i and δ_i^+ become larger (for Model 2), or we have more nonzero parameters a_{it} (in Model 1), so that more coefficients have to be stored in Constraints (1). In contrast, changing the size of the items (c_L vs. c_H) has almost no effect on the number of nonzeros, if the parameters are otherwise the same.

Altogether, we can summarize that the improved Model 2 seems to have the best properties in terms of complexity. To this end, we now use this formulation as a reference for the comparison with our new approach (i.e., Model 3, abbreviated by M3), see Tab. 7. Note that, again, we just selected a few subsets of Category (A), while the full details are shifted to Tab. C.12 in AppendixC.

				n_{var}		n_{con}		n_{nz}	
				M2	M3	M2	M3	M2	M3
n	$\bar{s}$	d_i	c_i	(Red)	(Red)	(Red)	(Red)	(Red)	(Red)
100	100	d_S	c_L	8.4	**5.2**	13.8	**13.4**	168.1	**160.7**
			c_H	8.3	**4.4**	14.1	**9.8**	173.5	**125.6**
		d_L	c_L	8.4	**5.2**	13.7	**13.4**	243.0	**232.8**
			c_H	8.3	**3.8**	13.7	**9.0**	249.7	**147.5**
	120	d_S	c_L	8.6	**5.2**	13.9	**13.6**	147.1	**144.0**
			c_H	8.6	**4.5**	13.7	**10.0**	145.1	**116.1**
		d_L	c_L	8.6	**5.2**	13.8	**13.6**	219.9	**213.1**
			c_H	8.7	**3.9**	13.5	**9.2**	219.3	**144.6**
200	200	d_S	c_L	33.4	**20.3**	54.9	**53.4**	742.4	**711.9**
			c_H	33.1	**18.7**	56.2	**41.0**	747.6	**607.6**
		d_L	c_L	33.4	**20.3**	55.3	**53.4**	1249.6	**1185.6**
			c_H	32.9	**17.2**	54.9	**39.2**	1227.0	**924.9**
	240	d_S	c_L	34.0	**20.3**	54.2	**53.9**	627.5	**622.9**
			c_H	34.4	**18.9**	54.2	**42.3**	634.3	**563.6**
		d_L	c_L	34.2	**20.3**	54.7	**54.2**	1063.9	**1049.2**
			c_H	34.1	**17.7**	55.0	**40.7**	1060.9	**852.0**

Table 7: Structural comparison involving the new formulations. The numbers listed in the table indicate units of 10^3, and the best number per category is printed in boldface.

We would like to especially mention the following observations:

- Obviously, for any of the three indicators, the reduced version of M2 is worse than the new compact formulation M3.
- In terms of variables, we see that the new job-to-job correspondence applied to construct Model 3 roughly causes (at least) an additional 40% of reduction compared to Model 2. Moreover, we see that the improvement is particularly successful for large item sizes (c_H), because then many pairs (i, k) do not appear in Δ_{red}, because they exceed the capacity when executed on the same server.
- Considering the number of constraints, we see that the difference between M2 and M3 is only significant for large item sizes, because then the capacity constraints (of M3) only have to be formulated for much fewer feasible pairs (i, k) (due to the incompatibility of the item pairs given by (R1*)).

5.3. Computational Results for Category (A)

After having dealt with some structural properties of the various ILP formulations in the previous subsection, we now investigate their practical performance in more details. To this end, we first study the computational behavior for the instances of Category (A), and we will use the results obtained from these experiments to later define a reasonable test environment for the second instance category. As stated earlier, the maximum solution time is set to 1800 seconds. Moreover, observe that whenever an instance could not be solved within this amount of time, we will use $t = 1800s$ (and the best feasible point obtained so far) within the respective averages. In Tab. 8, we report about the average computation times and the number of instances solved to proven optimality for the 32 subsets (each of which containing five instances) from Category (A). Due to the large benefits of applying the reduction methods (see Tab. 6), here we do not provide the results for the unimproved formulations. However, since the raw version of Model 1 represents the only competitor from the existing literature, we include an additional column termed '(Lit)' to give an overview of the results obtained in [4, Tab. 10] and, thus, enable a rough comparison to our proposed ILP formulations. Moreover, we will use boldface to indicate the best formulation. By *best* we mean that the respective approach could solve the largest number of instances, where ties are broken by the smaller time required.

Remark 7. *The average times in [4, Tab. 10] only included the data from the instances attempted successfully which differs from our strategy explained before. To facilitate the comparison, we therefore recalculated the values from the literature in a sense that each unsolved instance contributes with the time limit of* 1800 *seconds (which was also used in [4]).*

Among others, the following important observations can be made based on the results collected in Tab. 8:

- From an overall point of view, any of the new or improved formulations was better than the state-of-the-art model from the literature. In fact, any of our approaches could solve at least 10 instances more than the unimproved version of M1 (see column '(Lit)'). In particular, the improvements of Model 1 lead to a remarkable number of 22 additional instances that were solved to proven optimality.
- Only for four specific parameter constellations, the original model from [4] performed best. In two of these cases, however, note that also the reduced variant of M1 was able to solve all instances, but with a moderately larger average time. We attribute this observation to the fact that different computational environments have been used, and that, in addition, the solvers apply highly randomized solution strategies.
- Any of the assignment models showed the best performance for at least one subset of instances. The low complexity of M2 and M3 leads to very convincing results especially for the smaller instances with $n = 50$ items. In fact, M3 is the only compact formulation that is able to cope with all the 40 instances and, on top of that, required the smallest computation time for most of them. For medium instance sizes (with $n \in \{100, 150\}$ items), M1 is on average slightly better than the other formulations, but any of the approaches is the winner in four sub-categories of 5 instances each. Only for the largest instances (with $n = 200$) the better performance of M1 becomes a bit more apparent.

To get some further insights, we additionally report on the objective values obtained by the three formulations (for both, the ILP and the LP relaxation) in Tab. 9. Remember that, whenever an instances could not be solved to optimality within the given time limit, the best objective value found so far was used as an upper bound for the true optimal value. Based on Tab. 9, we highlight the following interesting results:

- First of all, let us state that any of the LP relaxations could be solved in (much) less than 400 seconds. Moreover, in all these calculations, the LP values of the three assignment

				M1 (Red)		M2 (Red)		M3 (Red)		(Lit) = [4]	
n	$\bar{s}$	d_i	c_i	t	*opt*	t	*opt*	t	*opt*	t	*opt*
50	50	d_S	c_L	7.4	(5)	6.3	(5)	**3.6**	**(5)**	36.6	(5)
			c_H	8.6	(5)	**2.8**	**(5)**	3.7	(5)	35.7	(5)
		d_L	c_L	367.8	(4)	71.9	(5)	**17.0**	**(5)**	1453.8	(1)
			c_H	12.3	(5)	5.0	(5)	**0.2**	**(5)**	481.5	(5)
	60	d_S	c_L	14.0	(5)	**3.3**	**(5)**	4.6	(5)	368.5	(4)
			c_H	37.8	(5)	363.1	(4)	**3.2**	**(5)**	412.3	(4)
		d_L	c_L	251.2	(5)	5.6	(5)	**1.1**	**(5)**	1125.4	(2)
			c_H	5.7	(5)	2.2	(5)	**0.3**	**(5)**	554.2	(4)
Average (Sum)				88.1	(39)	57.5	(39)	**4.2**	**(40)**	558.5	(30)
100	100	d_S	c_L	22.1	(5)	**4.2**	**(5)**	46.8	(5)	785.0	(3)
			c_H	738.6	(4)	956.7	(3)	1128.7	(3)	**479.3**	**(5)**
		d_L	c_L	1457.9	(1)	1800.0	(0)	**1447.4**	**(1)**	1800.0	(0)
			c_H	1500.4	(1)	1555.6	(1)	**1443.4**	**(1)**	1800.0	(0)
	120	d_S	c_L	**91.2**	**(5)**	152.7	(5)	381.8	(4)	855.4	(3)
			c_H	1042.8	(3)	1800.0	(0)	1800.0	(0)	**641.4**	**(5)**
		d_L	c_L	850.4	(3)	803.8	(3)	**635.2**	**(4)**	1800.0	(0)
			c_H	1481.0	(1)	1329.9	(2)	**733.3**	**(3)**	1800.0	(0)
Average (Sum)				**898.0**	**(23)**	1050.4	(19)	952.1	(21)	1245.1	(16)
150	150	d_S	c_L	462.4	(4)	**404.5**	**(4)**	601.1	(4)	659.4	(4)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
		d_L	c_L	**1502.3**	**(1)**	1800.0	(0)	1726.6	(1)	1800.0	(0)
			c_H	**1517.8**	**(1)**	1800.0	(0)	1628.7	(1)	1800.0	(0)
	180	d_S	c_L	**93.7**	**(5)**	377.7	(4)	634.1	(4)	290.0	(5)
			c_H	1682.6	(2)	1800.0	(0)	1800.0	(0)	**1283.7**	**(2)**
		d_L	c_L	1800.0	(0)	**1620.0**	**(1)**	1800.0	(0)	1800.0	(0)
			c_H	1800.0	(0)	**1649.1**	**(1)**	1797.6	(1)	1800.0	(0)
Average (Sum)				**1332.3**	**(13)**	1406.4	(10)	1473.5	(11)	1404.1	(11)
200	200	d_S	c_L	**582.8**	**(4)**	730.5	(3)	1595.9	(2)	1794.3	(1)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
		d_L	c_L	**1574.1**	**(1)**	1800.0	(0)	1800.0	(0)	1800.0	(0)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
	240	d_S	c_L	220.1	(5)	1083.7	(2)	1144.1	(2)	**180.1**	**(5)**
			c_H	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)
			c_H	1800.0	(0)	1800.0	(0)	1800.0	(0)	1800.0	(0)
Average (Sum)				**1422.1**	**(10)**	1576.8	(5)	1692.5	(4)	1596.8	(6)
Total: Average (Sum)				**935.2**	**(85)**	1022.8	(73)	1030.6	(76)	1201.1	(63)

Table 8: Number *opt* of instances solved to optimality and required (average) computation times t (in seconds) for Category (A).

				z			z_{LP}		
				M1	M2	M3	M1	M2	M3
n	$\bar{s}$	d_i	c_i	(Red)	(Red)	(Red)	(Red)	(Red)	(Red)
50	50	d_S	c_L	19.6	19.6	19.6	19.6	19.6	19.6
			c_H	25.8	25.8	25.8	25.6	25.6	25.6
		d_L	c_L	30.4	30.0	30.0	30.0	30.0	30.0
			c_H	46.0	46.0	46.0	46.0	46.0	46.0
	60	d_S	c_L	16.8	16.8	16.8	16.8	16.8	16.8
			c_H	24.0	24.0	24.0	23.6	23.6	23.6
		d_L	c_L	29.6	29.6	29.6	29.6	29.6	29.6
			c_H	41.6	41.6	41.6	41.6	41.6	41.6
Average				**29.2**	**29.2**	**29.2**	29.1	29.1	29.1
100	100	d_S	c_L	22.4	22.4	22.4	22.4	22.4	22.4
			c_H	34.8	35.4	35.4	33.6	33.6	33.6
		d_L	c_L	36.8	37.6	36.0	34.4	34.4	34.4
			c_H	54.0	51.6	51.0	49.2	49.2	49.2
	120	d_S	c_L	20.0	20.0	20.2	20.0	20.0	20.0
			c_H	30.6	31.0	31.0	27.2	27.2	27.2
		d_L	c_L	31.6	31.6	31.2	30.8	30.8	30.8
			c_H	47.8	46.6	45.6	44.4	44.4	44.4
Average				34.7	34.5	**34.1**	32.8	32.8	32.8
150	150	d_S	c_L	22.0	22.0	22.0	21.6	21.6	21.6
			c_H	53.8	42.8	46.8	34.0	34.0	34.0
		d_L	c_L	43.2	40.8	41.2	38.8	38.8	38.8
			c_H	105.2	62.4	58.6	52.8	52.8	52.8
	180	d_S	c_L	19.6	19.8	19.8	19.6	19.6	19.6
			c_H	89.4	39.2	42.2	28.0	28.0	28.0
		d_L	c_L	43.8	33.6	34.2	31.6	31.6	31.6
			c_H	116.6	59.2	57.2	47.6	47.6	47.6
Average				61.7	**40.0**	40.2	34.2	34.2	34.2
200	200	d_S	c_L	24.6	25.4	26.0	24.4	24.4	24.4
			c_H	247.2	49.6	59.0	31.6	31.6	31.6
		d_L	c_L	184.0	42.4	42.8	38.0	38.0	38.0
			c_H	317.2	78.0	71.8	53.6	53.6	53.6
	240	d_S	c_L	21.6	22.6	24.0	21.2	21.2	21.2
			c_H	163.2	51.4	58.0	30.4	30.4	30.4
		d_L	c_L	154.4	34.8	37.2	32.4	32.4	32.4
			c_H	320.4	83.2	66.0	46.0	46.0	46.0
Average				179.1	48.4	**48.1**	34.7	34.7	34.7

Table 9: Comparison of the average objective values z and z_{LP} for the integer models and their LP relaxations, respectively.

models were equivalent. Although we noticed in our theoretical part that any relations between the three LP values are possible (see Theorem 5), here we observed that they were always equal to (the integer value) $z^\star_{LP} = 2 \cdot h$. Note that this bound is easily implied by requiring h servers to be used, because any server will at least produce one fire-up.

- In many cases (especially for all scenarios with $n = 50$), all the assignment models perform equivalently, but they do not always identify the optimality of the feasible point found so far. For larger numbers of items (that is, $n \in \{150, 200\}$), we see that M1 partly struggles to find good integer solutions, particularly when large item sizes (c_H) are involved. In the latter case, the remaining ILP formulations lead to considerably better results. By way of example, not a single compact model could solve any of the c_H-instances for $n = 200$ (see Tab. 8), but M2 and M3 lead to much better feasible points[9]. We attribute this observation to the less complex model itself, because especially M3 highly benefits from the additional reductions (in terms of x-variables) caused by large item sizes, see (R1*).

Having a holistic view on Tab. 8 and Tab. 9, it seems that the general structure of M1 may be rather beneficial for branching because there are many interactions between the different types of variables, whereas the internal heuristics applied by Gurobi do not always perform successfully given the relatively large complexity of the formulation. For M2 and M3, the data suggest that it tends to be the other way around.

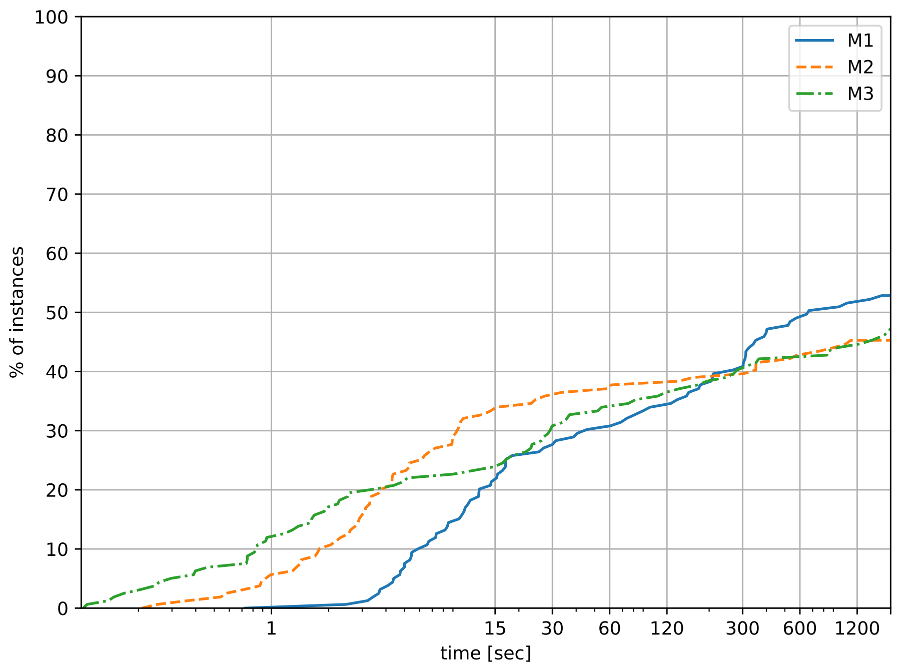


Figure 6: Performance profile for the three formulations applied to the instances of Category (A)

Although M1 represents the best compact formulation with respect to the criteria applied in Tab. 8, the general structure of M2 and M3 seems to be very appropriate to find near-optimal solutions of good approximation quality much quicker than M1 (at least for the instances considered here). Consequently, all models have their specific advantages and can be useful depending on which goal (e.g., large number of optimal solutions vs. good-quality feasible points in short times) shall be achieved. This impression is also supported by the performance profile depicted in

[9]A similar observation is also true for the d_L-instances.

Fig. 6. In it we see for each of the three models, which percentage of instances from Category (A) could be solved in which time. According to Fig. 6, the practical choice of the respective compact model for solving an instance should also depend on the available time frame. If only very little time is available to calculate a feasible solution (here: about up to four seconds), then Model 3 should be preferred, because on this short time scale (due to its very low complexity) it provides significantly better results than the other two models. If, on the other hand, decisions are allowed to take a medium amount of time (here: up to four minutes), Model 2 provides the best compromise between complexity and performance. Finally, the advantages of Model 1 become particularly clear in the long run.

5.4. Complexity of the ILP Formulations and Computational Results for Category (B)

In this section we would like to present a selection of the numerical results for the second category of instances (from [26]). Remember that these instances are characterized differently than those of Category (A), since there are only two main parameters influencing the shape of an instance. More precisely, as already stated at the beginning of Sect. 5, one of these parameters determines the number of active jobs per instant of time, while the other one more or less defines the temporal interaction between successive time steps. Contrary to the previous experiment, for a fixed number $|T|$ of time steps, we now consider ten classes (numbered by I-X) each of which consisting of 10 instances. To recapitulate, we repeat once again that the exact construction data of these classes are contained in [26, Tab. 1]. However, to give a rough overview, the following more detailed explanations can be made:

(a) Among the ten classes, Classes I-V and VII can be considered as relatively easy, especially because here a very high percentage of the jobs from the preceeding time step is overtaken for the current time step, so that we only have a few number of new jobs per instant of time. Moreover, the total number of active jobs per instant of time is not too large yet. Both these features lead to manageable cardinalities of I (and K). For the purpose of an internal ranking, it can be said that the instances referred to in the previous lines become (on average) more difficult with increasing class index.

(b) For Classes VI and IX, the b-parameters (from the construction described at the beginning of Sect. 5) are significantly smaller than for the other classes, meaning that we have (much) more new jobs per time step. This leads to a larger total number of jobs, i.e., to larger sets I and K, making the instances more difficult.

(c) For Classes VIII-X, the a-parameters (from the construction described at the beginning of Sect. 5) are larger than for the other classes. Hence, we can have a larger number of jobs per time step, so that again the sets I and K (and thus the instances) become more complex.

In a first test scenario, we collect the numerical data for all three (reduced) compact formulations, but only for those instances having a small number of time steps, i.e., for $|T| \in \{10, 15, 20\}$. Based on these results, it will be decided which of the models should be further investigated for larger instances. To this end, we particularly notice:

- The different levels of hardness (of the instance classes) described in (a)-(c) of the above list can be observed to a large extent for any of the three formulations.
- According to Tab. 10 (upper table), we see that M1 is able to solve the largest number of instances for any $|T| \in \{10, 15, 20\}$. From an overall point of view, M3 performs second best, but for many subsets it is able to solve the same number of instances as M1 in (slightly) less time. Regarding M2, it can be said that it typically leads to the worst results, because it can only solve a single subset the fastest (but performs only slightly better than M1 in this case). Altogether, these observations clearly support the general trends already observed for Category (A).

$\|T\|$	10						15						20					
	M1		M2		M3		M1		M2		M3		M1		M2		M3	
Class	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*
I	0.1	(10)	0.1	(10)	**0.0**	**(10)**	0.7	(10)	0.3	(10)	**0.0**	**(10)**	1.0	(10)	0.6	(10)	**0.1**	**(10)**
II	0.2	(10)	0.2	(10)	**0.1**	**(10)**	0.7	(10)	0.4	(10)	**0.2**	**(10)**	1.4	(10)	1.0	(10)	**0.5**	**(10)**
III	0.7	(10)	1.0	(10)	**0.1**	**(10)**	1.4	(10)	0.9	(10)	**0.2**	**(10)**	3.5	(10)	2.9	(10)	**0.6**	**(10)**
IV	1.1	(10)	2.0	(10)	**0.3**	**(10)**	5.8	(10)	5.4	(10)	**1.3**	**(10)**	10.8	(10)	10.7	(10)	**2.5**	**(10)**
V	2.0	(10)	3.4	(10)	**0.5**	**(10)**	10.6	(10)	43.8	(10)	**3.8**	**(10)**	61.0	(10)	202.6	(9)	**22.2**	**(10)**
VI	7.5	(10)	9.1	(10)	**3.9**	**(10)**	20.9	(10)	56.7	(10)	**12.4**	**(10)**	37.5	(10)	**34.5**	**(10)**	50.5	(10)
VII	2.5	(10)	3.3	(10)	**0.6**	**(10)**	10.5	(10)	11.2	(10)	**2.3**	**(10)**	31.1	(10)	94.2	(10)	**25.9**	**(10)**
VIII	196.0	(9)	552.6	(7)	**184.9**	**(9)**	602.6	(7)	1102.3	(4)	**580.4**	**(7)**	**850.5**	**(8)**	1481.6	(2)	1135.7	(4)
IX	**38.0**	**(10)**	197.1	(9)	187.3	(9)	**324.1**	**(9)**	646.2	(7)	586.1	(7)	**650.4**	**(7)**	1122.3	(4)	975.8	(5)
X	33.0	(10)	57.4	(10)	**2.6**	**(10)**	573.7	(7)	755.5	(6)	**552.6**	**(7)**	**800.9**	**(6)**	1107.8	(5)	858.5	(6)
Average (Sum)	**28.1**	**(99)**	82.6	(96)	38.0	(98)	**155.1**	**(93)**	262.3	(87)	173.9	(91)	**244.8**	**(91)**	405.8	(80)	307.2	(85)

$\|T\|$	15									20								
	n_{var}			n_{con}			n_{nz}			n_{var}			n_{con}			n_{nz}		
Class	M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
I	0.9	0.6	**0.2**	1.5	1.0	**0.6**	6.1	5.2	**2.9**	1.3	0.9	**0.4**	2.2	1.4	**0.8**	9.5	8.0	**4.7**
II	1.6	1.2	**0.6**	2.6	2.4	**1.2**	13.1	17.0	**8.2**	2.5	1.8	**0.9**	3.9	3.5	**1.8**	20.5	26.7	**13.9**
III	2.2	1.7	**0.7**	3.4	3.5	**1.5**	19.3	29.4	**11.0**	3.2	2.4	**1.1**	4.9	4.9	**2.3**	29.9	44.8	**18.9**
IV	3.1	2.4	**1.1**	4.5	5.2	**2.3**	28.8	52.6	**20.8**	4.5	3.5	**1.8**	6.7	7.6	**3.6**	45.0	83.8	**36.5**
V	3.9	3.1	**1.5**	5.6	7.2	**3.0**	38.3	81.9	**30.0**	5.7	4.5	**2.3**	8.2	10.2	**4.6**	59.6	127.0	**51.6**
VI	9.8	8.5	**5.7**	12.5	22.2	**9.5**	**78.4**	309.7	145.2	15.6	13.6	**9.6**	19.8	35.4	**15.8**	**131.2**	534.6	271.2
VII	3.9	3.1	**1.5**	5.6	7.0	**3.0**	37.9	79.3	**29.3**	5.7	4.5	**2.3**	8.2	10.1	**4.5**	58.9	124.7	**50.8**
VIII	5.3	4.3	**2.4**	7.3	10.5	**4.5**	**48.9**	127.2	50.8	8.1	6.7	**3.9**	11.1	16.0	**7.1**	**78.2**	211.7	92.5
IX	10.2	8.9	**6.1**	12.9	23.4	**10.1**	**80.5**	327.4	157.3	16.2	14.1	**10.1**	20.4	36.9	**16.6**	**134.6**	556.4	288.7
X	6.3	5.3	**2.9**	8.5	13.0	**5.4**	**60.7**	179.6	68.5	9.5	7.9	**4.6**	12.8	19.3	**8.4**	**97.1**	290.8	121.6
Average	4.7	3.9	**2.3**	6.4	9.5	**4.1**	**41.2**	120.9	52.4	7.2	6.0	**3.7**	9.8	14.5	**6.6**	**66.5**	200.8	95.0

Table 10: Number *opt* of instances solved to optimality and required average times in seconds (upper table) as well as structural properties in units of 10^3 (lower table) for instances from Category (B) having small number of time steps

- Contrary to the previous experiments, we see that the structural properties of M2 are much worse than those of M1 and M3, see Tab. 10 (lower table). More precisely, especially for the harder instance classes (i.e., for Classes VI and VIII-X, see part (b) and (c) of the above list), significant differences in terms of the numbers of constraints and nonzeros can be noticed. To better understand this, we have to remember that, by way of example, the numbers of constraints in the unimproved versions of M1 and M2 resulted to $n \cdot (|T| + 1 + |T_S|) + n^2$ and $3n^2 + n$ (see Sect. 2), respectively, and that the effects of the reductions were (approximately) comparably strong (not precisely the same, but on a similar level). Since the total number of jobs n is particularly large for the (difficult) instance classes addressed in the current discussion (and since we only consider a rather small number $|T|$ of time steps), we typically have $|T_S| \leq |T| \ll n$. With this in mind, the coefficient of the quadratic term (in the number of constraints) is generally larger for M2. Of course, this large number of constraints then also influences the number of nonzeros.

Although there is no separate table for this here, we mention for the sake of completeness that the optimal LP values were always identical for each of the three models. Furthermore, contrary to the results from Tab. 9, there were no significant differences as regards the objective values of the best integer solution found. Most likely, the instances (with only a few time steps) are not yet difficult enough to observe this behavior again.

To conclude this introductory investigation for the instances of Category (B), let us have a look at a performance profile, see Fig. 7, summarizing the numerical results for the 300 instances attempted so far. Similar to Category (A), we again see that M3 (usually having the least complex model structure) performs best when decisions have to be made very quickly. Contrary to the situation depicted in Fig. 6, now we see that M1 and M2 perform almost equivalently even for the smaller solution times which is a clear indicator that the instances from Category (B) are rather not favorable to M2. Again, in the long run, the numerical advantages of M1 (also compared to M3) become visible.

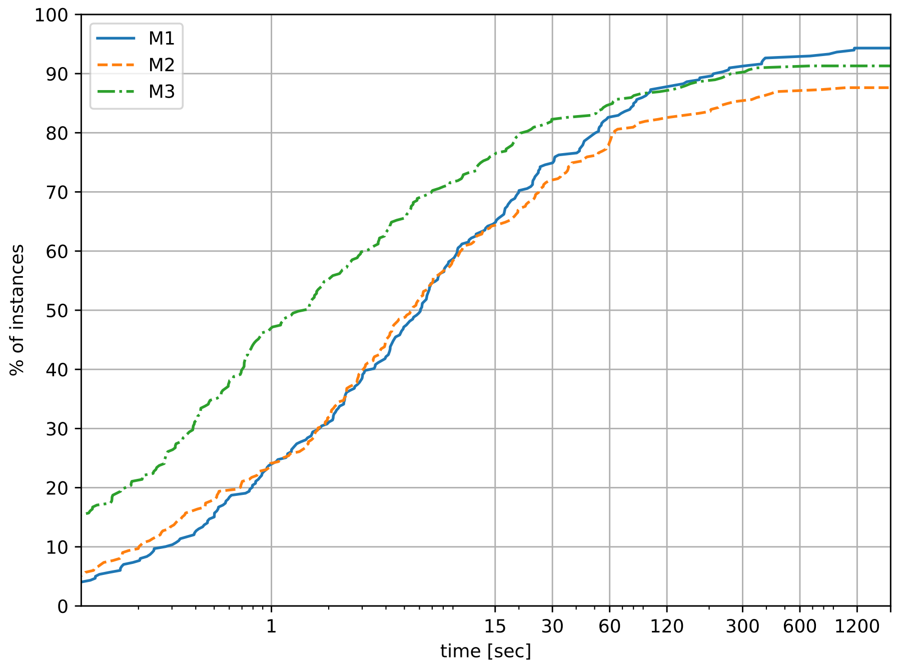


Figure 7: Performance profile for the three formulations applied to instances of Category (B) with $|T| \in \{10, 15, 20\}$

Thanks to the aforementioned results, we now only consider M1 and M3 for growing instance sizes. To be more precise, our final experiment deals with the medium-sized sets of time steps $|T| \in \{30, 40, 50, 60\}$. Note that, for the ordinary TBPP, already these instances have only been tackled heuristically in [26], so that not going for even larger instance sizes from Category (B) can be justified. Here we only provide two overview tables (see Tab. 11) summarizing the main results of our computational study. We can see that, for each fixed $|T|$, M1 is again superior to M3 in terms of the number of instances solved to optimality (see upper table in Tab. 11). With increasing number $|T|$ of time steps, we also observe that M1 becomes more and more convincing even in the rather easy instance classes which were dominated by M3 before. Contrary to this, both approaches really struggle to find optimal solutions for the very hard Classes VIII-X. However, as shown in the lower table of Tab. 11, especially for these difficult instances M3 is again able to find feasible solutions of (much) better quality. The fact that both models offer their specific advantages also for the current set of 400 instances is underlined in Fig. 8. Again, we see that M3 can solve more instances in short time, but that the "distance" to M1 (compared to the previous figures) has become closer. In the long run (here from about 180 seconds on) the generally better performance of M1 becomes noticeable again.

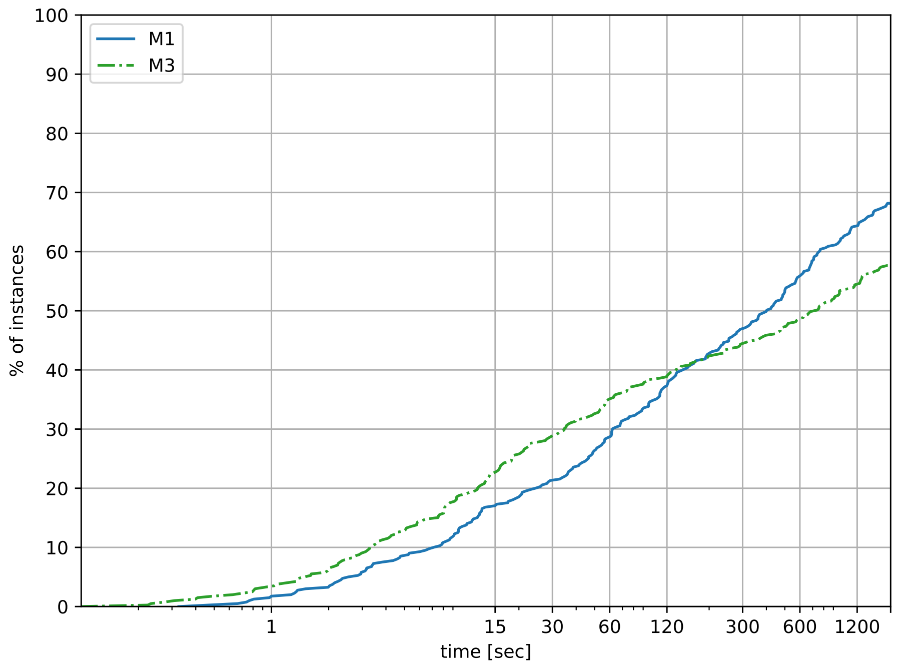


Figure 8: Performance profile for two compact formulations applied to instances of Category (B) with $|T| \in \{30, 40, 50, 60\}$

6. Conclusions

In this article, we studied compact formulations for a generalization of the temporal bin packing problem. More precisely, besides the mere number of servers required, also the sustainability of the operation mode (measured by the number of fire-ups) is contained in the overall optimization problem. At first, we propose a wide variety of reduction methods for the two existing ILP formulations from the literature, and show that they can lead to improved LP values and less complex compact models. Moreover, an alternative approach having even fewer variables and constraints is introduced, and the applicability of reduction methods is discussed also with respect to this new framework. From a theoretical point of view, one of our main contributions

$\|T\|$	30				40				50				60			
Class	M1		M3		M1		M3		M1		M3		M1		M3	
	t	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*	*t*	*opt*
I	1.7	(10)	**0.5**	**(10)**	73.0	(10)	**1.9**	**(10)**	32.9	(10)	**4.4**	**(10)**	16.6	(10)	**9.6**	**(10)**
II	4.6	(10)	**2.9**	**(10)**	9.5	(10)	**6.7**	**(10)**	25.7	(10)	**23.3**	**(10)**	**41.2**	**(10)**	65.9	(10)
III	15.6	(10)	**3.4**	**(10)**	212.3	(9)	**191.7**	**(9)**	**77.1**	**(10)**	182.3	(10)	**190.7**	**(10)**	318.5	(10)
IV	32.7	(10)	**13.0**	**(10)**	**115.6**	**(10)**	254.0	(9)	**309.3**	**(10)**	575.1	(10)	**557.7**	**(10)**	1280.6	(6)
V	408.3	(9)	**279.0**	**(9)**	**482.7**	**(9)**	554.7	(9)	**913.7**	**(9)**	1101.2	(7)	**1278.8**	**(5)**	1779.8	(1)
VI	**119.4**	**(10)**	459.3	(10)	**605.7**	**(10)**	1008.0	(6)	**842.2**	**(9)**	1672.0	(1)	**1360.6**	**(6)**	1715.6	(1)
VII	94.8	(10)	**33.8**	**(10)**	601.4	(9)	**506.3**	**(9)**	**1113.0**	**(6)**	1243.7	(6)	**1441.5**	**(5)**	1800.0	(0)
VIII	**1356.4**	**(4)**	1494.1	(2)	1681.3	(1)	**1667.3**	**(1)**	**1800.0**	**(0)**	1800.0	(0)	**1800.0**	**(0)**	1800.0	(0)
IX	**1386.5**	**(4)**	1662.4	(2)	**1530.9**	**(2)**	1800.0	(0)	**1670.7**	**(1)**	1800.0	(0)	**1789.9**	**(1)**	1800.0	(0)
X	**1416.9**	**(4)**	1448.6	(3)	**1800.0**	**(0)**	1800.0	(0)	**1800.0**	**(0)**	1800.0	(0)	**1800.0**	**(0)**	1800.0	(0)
Average (Sum)	**483.7**	**(81)**	539.7	(76)	**711.2**	**(70)**	779.1	(63)	**858.5**	**(65)**	1020.2	(54)	**1027.7**	**(57)**	1237.0	(38)

$\|T\|$	30		40		50		60	
Class	M1	M3	M1	M3	M1	M3	M1	M3
I	16.0	16.0	16.7	16.7	17.5	17.5	17.5	17.5
II	23.0	23.0	24.0	24.0	24.4	24.4	24.4	24.4
III	32.2	32.2	32.7	32.7	32.6	32.6	32.6	32.6
IV	35.2	35.2	35.4	35.5	36.0	36.0	36.2	36.8
V	40.6	40.5	41.9	41.9	43.3	43.3	65.6	47.8
VI	44.8	44.8	44.8	46.9	47.8	98.5	83.0	101.1
VII	40.4	40.4	40.9	40.9	42.8	42.4	45.1	47.7
VIII	47.2	48.6	50.6	55.8	73.0	65.2	77.9	78.3
IX	48.4	50.9	51.1	59.5	124.5	72.4	348.6	142.9
X	55.0	56.4	65.9	65.5	148.7	76.8	195.3	113.6
Average	**38.3**	38.8	**40.4**	41.9	59.1	**50.9**	92.6	**64.3**

Table 11: Upper table: Number of instances solved to optimality and average solution times (upper table) as well as best objective values found (lower table) for medium-sized instances of Category (B)

consists in proving that none of these three models is superior to another in terms of the quality of its LP bound. Finally, all approaches are numerically compared using different categories of test instances. It turns out that the reduction methods are very successful (depending on the model, the reductions in terms of variables and constraints are roughly 50% or even beyond) and therefore, from an overall point of view, each of the models shows a better performance than the state-of-the-art in the literature. On average, the improved version of Model 1 shows the most convincing results, while M2 and M3 (due to their low complexity) are particularly suitable for quickly finding high-quality feasible solutions. Altogether, the numerical tests thus underline the fact that each of the models has its individual areas of application.

In our future research we mainly want to deal with alternative modeling approaches for the TBPP-FU. In particular, we will focus on flow-based formulations, whose general strengths have recently been brilliantly summarized in [25], and exponential formulations, which can then be solved using branch-and-price. Note, however, that an efficient implementation of both approaches is challenging and not necessarily straightforward. On the one hand, the different states of an arcflow graph must not only contain information about the currently used capacity but also about the temporal dimension (i.e., at least the next terminating item or even all items forming the current state), thus making a smart construction of nodes and arcs more complicated. On the other hand, any branch-and-price framework offers a lot of tuning possibilities (e.g., branching schemes, heuristics, solving the pricing problems, etc.), so that a thorough analysis of all these ingredients has to be conducted. Since both of these alternative approaches require much more preparatory work, these topics could not be dealt within the scope of this already very detailed research article.

Conflict of Interest

The authors declare that they have no conflict of interest.

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] Arjona, J., Chatzipapas, A., Fernandez Anta, A., Mancuso, V.: A measurement-based analysis of the energy consumption of data center servers. Proceedings of the 5th International Conference on Future Energy System, 63–74 (2014)

[3] Arkin, E.M., Silverberg, E.B.: Scheduling jobs with fixed start and end times. Discrete Applied Mathematics 18(1), 1–8 (1987)

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

[5] Bacci, T., Frangioni, A., Gentile, C.: Start-Up/Shut-Down MINLP Formulations for the Unit Commitment with Ramp Constraints. Research Report 20-02, Istituto di Analisi dei Sistemi ed Informatica 'A. Ruberti', Consiglio Nazionale delle Ricerche (2020) (`http://www.iasi.cnr.it/new/publications.php/id_p/2/anno/0/id_autore/156/id_tipologia/6/rep/4789`)

[6] Bacci, T., Frangioni, A., Gentile, C., Tavlaridis-Gyparakis, K.: New MINLP Formulations for the Unit Commitment Problems with Ramping Constraints. Research Report 19-04, Istituto di Analisi dei Sistemi ed Informatica 'A. Ruberti', Consiglio Nazionale delle Ricerche (2019) (`http://www.optimization-online.org/DB_HTML/2019/10/7426.html`)

[7] Baldi, M.M., Manerba, D., Perboli, G., Tadei, R.: A Generalized Bin Packing Problem for parcel delivery in last-mile logistics. European Journal of Operational Research 274(3), 990-999 (2019)

[8] Barnett Jr., T., Jain, S., Andra, U., Khurana, T.: Cisco Visual Networking Index (VNI) Complete Forecast Update, 2017–2022. APJC Cisco Knowledge Network (CKN) Presentation (2018) (`https://www.cisco.com/c/dam/m/en_us/network-intelligence/service-provider/digital-transformation/knowledge-network-webinars/pdfs/1213-business-services-ckn.pdf`)

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

[10] Belov, G., Scheithauer, G.: A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. European Journal of Operational Research 171(1), 85–106 (2006)

[11] Boschetti, M.A., Hadjiconstantinou, E., Mingozzi, A.: New upper bounds for the two-dimensional othogonal non guillotine cutting stock problem. IMA Journal of Management Mathematics 13(2), 95–119 (2002)

[12] Brandão, F., Pedroso, J.P.: Bin packing and related problems: General arc-flow formulation with graph compression. Computers & Operations Research 69, 56–67 (2016)

[13] Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems 25(6), 599–616 (2009)

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

[15] Caprara, A., Furini, F., Malaguti, E., Traversi, E.: Solving the temporal knapsack problem via recursive Dantzig-Wolfe reformulation. Information Processing Letters, 116(5), 379–386 (2016)

[16] Caprara, A., Toth, P.: Lower bounds and algorithms for the 2-dimensional vector packing problem. Discrete Applied Mathematics 111(3), 231–262 (2001)

[17] Clautiaux, F., Carlier, J., Moukrim, A.: New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. Computers & Operations Research 34(8), 2223–2250 (2007)

[18] Coffman Jr., E.G., Csirik, J., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: Survey and classification. In Pardalos, P.M., Du, D., Graham, R.L. (eds), Handbook of Combinatorial Optimization, 455–531, Springer, New York (2013)

[19] Coffman Jr., E.G., Garey, M.R., Johnson, D.S.: Approximation Algorithms for Bin Packing – An Updated Survey. In: Ausiello, G., Lucertini, M., Serafini, P. (eds), Algorithm Design for Computer System Design. International Centre for Mechanical Sciences (Courses and Lectures), vol. 284, Springer, Vienna (1984)

[20] Cohen, M.C., Keller, P.W., Mirrokni, V., Zadimoghaddam, M.: Overcommitment in Cloud Services: Bin Packing with Chance Constraints. Management Science 65(7), 3255–3271 (2019)

[21] Côté, J.F., Dell'Amico, M., Iori, M.: Combinatorial Benders' cuts for the strip packing problem. Operations Research 62(3), 643–661 (2014)

[22] Crainic, T.G., Gobbato, L., Perboli, G., Rei, W.: Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging meta-heuristic. European Journal of Operational Research 253(2), 404–417 (2016)

[23] Dargie, W.: A stochastic model for estimating the power consumption of a server. IEEE Transactions on Computers 64(5), 1311–1322 (2015)

[24] de Cauwer, M., Mehta, D., O'Sullivan, B.: 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, 157–164, (2016)

[25] de Lima, V.L., Alves, C., Clautiaux, F., Iori, M., Valeério de Carvalho, J.M.: Arc Flow Formulations Based on Dynamic Programming: Theoretical Foundations and Applications. arXiv preprint (2020) (`https://arxiv.org/abs/2010.00558`)

[26] Dell'Amico, M., Furini, F., Iori, M.: A Branch-and-Price Algorithm for the Temporal Bin Packing Problem. Computers & Operations Research 114, Article 104825 (2020)

[27] Delorme, M., Iori, M.: Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. INFORMS Journal on Computing 32(1), 101–119 (2020)

[28] Delorme, M. Iori, M., Martello, S.: Bin packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms. Research Report OR-15-1, University of Bologna (2015)

[29] Delorme, M. Iori, M., Martello, S.: Bin packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms. European Journal of Operational Research 255, 1–20 (2016)

[30] Dósa, G., Li, R., Han, X., Tuza, Z.: Tight absolute bound for First Fit Decreasing bin packing: $FFD(L) \leq 11/9 \cdot OPT(L) + 6/9$. Theoretical Computer Science 510, 13–61 (2013)

[31] 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. Proceedings of the IEEE 107(1), 204–231 (2019)

[32] Frangioni, A., Gentile, C.: Solving Nonlinear Single-Unit Commitment Problems with Ramping Constraints. Operations Research 54, 767–775 (2006)

[33] Gilmore, P.C., Gomory, R.E.: A Linear programming approach to the cutting-stock problem (Part I). Operations Research 9, 849–859 (1961)

[34] Gschwind, T., Irnich, S.: Dual Inequalities for Stabilized Column Generation Revisited. INFORMS Journal on Computing 28(1), 175–194 (2016)

[35] Heßler, K., Gschwind, T., Irnich, S.: Stabilized branch-and-price algorithms for vector packing problems. European Journal of Operational Research 271(2), 401–419 (2018)

[36] Iori, M., de Lima, V.L., Martello, S., Miyazawa, F.K., Monaci, M.: Exact solution techniques for two-dimensional cutting and packing. *to appear in:* European Journal of Operational Research (2020) (`https://doi.org/10.1016/j.ejor.2020.06.050`)

[37] Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-Case Performance Bounds for Simple One-dimensional Packing Algorithms. SIAM Journal on Computing 3(4), 299-325 (1974)

[38] Jones, N.: How to stop data centres from gobbling up the world's electricity. Nature 561, 163–166 (2018)

[39] Kantorovich, L.V.: Mathematical methods of organising and planning production. Management Science 6, 366–422 (1939 Russian, 1960 English)

[40] Kaplan, J.M., Forrest, W., Kindler, N.: Revolutionizing data center energy efficiency. Technical report, McKinsey & Company (2008)

[41] Kenmochi, M., Imamichi, T., Nonobe, K., Yagiura, M., Nagamochi, H.: Exact algorithms for the two-dimensional strip packing problem with and without rotations. European Journal of Operational Research 198(1), 73–83 (2009)

[42] Koomey, J.: Worldwide electricity used in data centers, Environmental Research Letters 3, 1–8 (2008)

[43] Lodi, A., Martello, S., Monaci, M.: Two-dimensional packing problems: A survey. European Journal of Operational Research 141(2), 241–252 (2002)

[44] López-Pires, F., Barán, B.: Virtual Machine Placement Literature Review. arXiv Preprint (2015) (`http://arxiv.org/abs/1506.01509`)

[45] Luo, J.-P., Li, X., Chen, M.-R.: Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers. Expert Systems with Applications 41(13), 5804–5816 (2014)

[46] Manvi, S.S., Krishna Shyam, G.: Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. Journal of Network and Computer Applications 41, 424–440 (2014)

[47] Martello, S., Monaci, M., Vigo, D.: An exact approach to the strip-packing problem. INFORMS Journal on Computing 15(3), 310–319 (2003)

[48] Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, Chichester, New York (1990)

[49] Martinovic, J., Delorme, M., Iori, M., Scheithauer, G., Strasdat, N.: Improved flow-based formulations for the skiving stock problem. Computers & Operations Research 113, Article 104770 (2020)

[50] Martinovic, J., Hähnel, M., Scheithauer, G., Dargie, W., Fischer, A.: Cutting Stock Problems with Nondeterministic Item Lengths: A New Approach to Server Consolidation. 4OR 17(2), 173–200 (2019)

[51] Martinovic, J., Scheithauer, G., Valério de Carvalho, J.M.: A Comparative Study of the Arcflow Model and the One-Cut Model for one-dimensional Cutting Stock Problems. European Journal of Operational Research 266(2), 458–471 (2018)

[52] Martinovic, J., Selch, M.: Mathematical Models and Approximate Solution Approaches for the Stochastic Bin Packing Problem. Preprint MATH-NM-02-2020, Technische Universität Dresden (2020) (`http://www.optimization-online.org/DB_HTML/2020/09/8043.html`)

[53] Möbius, C., Dargie, W., Schill, A.: Power consumption estimation models for servers, virtual machines, and servers. IEEE Transactions on Parallel and Distributed Systems 25(6), 1600–1614 (2014)

[54] Scheithauer, G.: Introduction to Cutting and Packing Optimization – Problems, Modeling Approaches, Solution Methods. International Series in Operations Research & Management Science 263, Springer, 1.Edition (2018)

[55] Spieksma, F.C.R.: A branch-and-bound algorithm for the two-dimensional vector packing problem. Computers and Operations Research 21, 19–25 (1994)

[56] Valério de Carvalho, J.M: Exact solution of bin packing problems using column generation and branch-and-bound. Annals of Operations Research 86, 629–659 (1999)

[57] Valério de Carvalho, J.M.: LP models for bin packing and cutting stock problems. European Journal of Operations Research 141(2), 253–273 (2002)

[58] Valério de Carvalho, J.M.: Using Extra Dual Cuts to Accelerate Column Generation. INFORMS Journal on Computing 17(2), 175–182 (2005)

[59] van Ackooij, W., Danti Lopez, I., Frangioni, A., Lacalandra, F., Tahanan, M.: Large-scale unit commitment under uncertainty: an updated literature survey. Annals of Operations Research 271, 11–85 (2018)

[60] Vance, P.: Branch-and-price algorithms for the one-dimensional cutting stock problem. Computational Optimization and Applications 9, 211–228 (1998)

[61] Vance, P., Barnhart, C., Johnson, E.L., Nemhauser, G.L.: Solving binary cutting stock problems by column generation and branch-and-bound. Computational Optimization and Applications 3(2), 111–130 (1994)

AppendixA. Detailed Solutions for Example 1

- **(Lit):** $z_1 = z_2 = 1$ and

x_{ik}	1	2	3	4
1	1/2	1/2	0	0
2	2/3	1/3	0	0
3	2/3	1/3	0	0
4	7/9	2/9	0	0

y_{tk}	1	2	3	4
1	1	2/3	0	0
2	1	1/3	0	0
3	1	1/3	0	0
4	0	0	0	0

w_{tk}	1	2	3	4
1	1	2/3	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

- **(Lit)** + **(R1):** $z_1 = z_2 = 1$ and

x_{ik}	1	2	3	4
1	1	-	-	-
2	1/3	2/3	-	-
3	2/3	1/3	0	-
4	7/9	2/9	0	0

y_{tk}	1	2	3	4
1	1	2/3	-	-
2	1	0	-	-
3	1	1/3	0	0
4	0	0	0	0

w_{tk}	1	2	3	4
1	1	2/3	-	-
3	0	1/3	0	0

- **(Lit)** + **(R2):** $z_1 = z_2 = 1$ and

x_{ik}	1	2	3	4
1	1	0	0	0
2	1/3	2/3	0	0
3	2/3	1/3	0	0
4	7/9	2/9	0	0

y_{tk}	1	2	3	4
1	1	2/3	0	0
2	1	0	0	0
3	1	1/3	0	0
4	0	0	0	0

w_{tk}	1	2	3	4
1	1	2/3	0	0
2	0	0	0	0
3	0	1/3	0	0
4	0	0	0	0

- **(Lit)** + **(R3):** $z_1 = z_2 = 1$ and

x_{ik}	1	2	3	4
1	1/3	2/3	0	0
2	2/3	1/3	0	0
3	0	1	0	0
4	1	0	0	0

y_{tk}	1	2	3	4
1	1	1	0	0
2	1	1	0	0
3	1	1	0	0
4	0	0	0	0

w_{tk}	1	2	3	4
1	1	1	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

- **(Lit) + (R1)-(R3):** $z_1 = z_2 = 1$ and

x_{ik}	1	2	3	4
1	1	-	-	-
2	0	1	-	-
3	0	1	0	-
4	1	0	0	0

y_{tk}	1	2	3	4
1	1	1	-	-
2	1	0	-	-
3	1	1	0	0
4	0	0	0	0

w_{tk}	1	2	3	4
1	1	1	-	-
3	0	1	0	0

AppendixB. Detailed Solutions for the Instances from Fig. 5

For the leftmost instance in Fig. 5, we obtain the following solutions of the LP relaxations:

- **Model 1:** Optimal value $z_{LP}^{(1),\star} = 5$ given by $z_1 = z_2 = 1$ and

x_{ik}	1	2	3
1	1	-	-
2	1	0	-
3	0	1	0

y_{tk}	1	2	3
1	1	-	-
2	0	-	-
3	1	1	0
4	0	0	0

w_{tk}	1	2	3
1	1	-	-
3	1	1	0

- **Model 2:** Optimal value $z_{LP}^{(2),\star} = 4.5$ given by $z_1 = z_2 = 1$ and

x_{ik}	1	2	3
1	1	-	-
2	1/2	1/2	-
3	1/2	1/2	0

w_{tk}	1	2	3
1	1	-	-
3	1/2	1	0

- **Model 3:** Optimal value $z_{LP}^{(3),\star} = 4$ given by

x_{ik}	1	2	3
1	1	-	-
2	1/3	2/3	-
3	2/3	-	1/3

i	1	2	3
w_i	1	2/3	1/3

For the middle instance in Fig. 5, we obtain the following solutions of the LP relaxations:

- **Model 1:** Optimal value $z_{LP}^{(1),\star} = 4.5$ given by $z_1 = z_2 = 1$ and

x_{ik}	1	2	3	4
1	1	-	-	-
2	1	0	-	-
3	0	1	0	-
4	1/2	1/2	0	0

y_{tk}	1	2	3	4
1	1	1	0	-
2	1	0	0	-
3	1	1/2	0	0
4	0	0	0	0

w_{tk}	1	2	3	4
1	1	1	0	-
3	0	1/2	0	0

- **Model 2:** Optimal value $z_{LP}^{(2),\star} = 4$ given by $z_1 = z_2 = 1$ and

x_{ik}	1	2	3	4
1	1	-	-	-
2	3/4	1/4	-	-
3	1/8	7/8	0	-
4	5/8	3/8	0	0

w_{tk}	1	2	3	4
1	1	7/8	0	-
3	0	1/8	0	0

- **Model 3:** Optimal value $z_{LP}^{(3),\star} = 4.5$ given by

x_{ik}	1	2	3	4
1	1	-	-	-
2	1	0	-	-
3	-	-	1	-
4	1/2	-	1/4	1/4

i	1	2	3	4
w_i	1	0	1	1/4

For the rightmost instance in Fig. 5, we obtain the following solutions of the LP relaxations:

- **Model 1:** Optimal value $z_{LP}^{(1),\star} = 4$ given by $z_1 = z_2 = 1$ and

x_{ik}	1	2	3
1	1	-	-
2	1/3	2/3	-
3	2/3	1/3	0

y_{tk}	1	2	3
1	1	2/3	-
2	1	-	-
3	1	1	0
4	0	0	0

w_{tk}	1	2	3
1	1	2/3	-
3	0	1/3	0

- **Model 2:** Optimal value $z_{LP}^{(2),\star} = 13/3$ given by $z_1 = z_2 = 1$ and

x_{ik}	1	2	3
1	1	-	-
2	1/3	2/3	-
3	2/3	1/3	0

w_{tk}	1	2	3
1	1	1	-
3	1/3	0	0

- **Model 3:** Optimal value $z_{LP}^{(3),\star} = 5$ given by

x_{ik}	1	2	3
1	1	-	-
2	-	1	-
3	1	-	0

i	1	2	3
w_i	1	1	1

AppendixC. Further Numerical Results

				n_{var}					n_{con}					n_{nz}				
				M1		M2		M3	M1		M2		M3	M1		M2		M3
n	$\bar{s}$	d_i	c_i	(Lit)	(Red)	(Lit)	(Red)	(Red)	(Lit)	(Red)	(Lit)	(Red)	(Red)	(Lit)	(Red)	(Lit)	(Red)	(Red)
50	50	d_S	c_L	7.8	3.7	4.2	2.2	**1.3**	12.1	6.8	7.6	3.5	**3.4**	102.3	49.1	95.9	35.0	**34.1**
			c_H	7.9	3.7	4.1	2.1	**1.0**	12.1	6.8	7.6	3.5	**2.3**	96.6	47.6	88.4	34.1	**21.1**
		d_L	c_L	8.9	3.9	4.2	2.2	**1.3**	13.7	7.6	7.6	**3.2**	3.4	156.8	67.4	126.5	**36.9**	40.2
			c_H	8.8	4.0	4.3	2.2	**0.7**	13.7	7.6	7.6	3.2	**1.8**	159.4	67.7	127.7	37.7	**16.7**
	60	d_S	c_L	7.9	3.8	4.3	2.3	**1.3**	12.4	7.0	7.6	**3.5**	**3.5**	92.5	46.1	82.9	32.4	**32.2**
			c_H	8.3	3.8	4.3	2.2	**1.0**	12.9	7.1	7.6	3.6	**2.5**	100.8	48.4	84.6	33.6	**21.8**
		d_L	c_L	8.8	4.0	4.3	2.2	**1.3**	13.7	7.7	7.6	**3.3**	3.5	154.2	67.1	124.8	**37.9**	40.9
			c_H	8.9	4.0	4.2	2.2	**0.8**	13.8	7.7	7.6	3.3	**2.1**	153.1	67.4	121.9	38.0	**19.2**
100	100	d_S	c_L	28.9	13.5	16.7	8.4	**5.2**	45.0	24.0	30.2	13.8	**13.4**	403.2	199.4	410.9	168.1	**160.7**
			c_H	29.1	13.5	16.3	8.3	**4.4**	45.0	24.1	30.2	14.1	**9.8**	408.2	203.4	412.7	173.5	**125.6**
		d_L	c_L	31.3	14.4	16.5	8.4	**5.2**	48.4	26.6	30.2	13.7	**13.4**	702.5	325.0	676.2	243.0	**232.8**
			c_H	30.9	14.3	16.3	8.3	**3.8**	47.6	26.4	30.2	13.7	**9.0**	702.3	329.7	686.8	249.7	**147.5**
	120	d_S	c_L	30.7	14.2	17.0	8.6	**5.2**	48.0	25.5	30.2	13.9	**13.6**	390.7	193.5	357.6	147.1	**144.0**
			c_H	30.8	14.2	17.0	8.6	**4.5**	48.1	25.6	30.2	13.7	**10.0**	387.0	194.4	352.8	145.1	**116.1**
		d_L	c_L	32.1	14.7	16.8	8.6	**5.2**	49.9	27.2	30.2	13.8	**13.6**	639.0	304.9	577.6	219.9	**213.1**
			c_H	32.8	15.0	17.0	8.7	**3.9**	51.2	27.7	30.2	13.5	**9.2**	678.7	320.3	597.6	219.3	**144.6**
150	150	d_S	c_L	63.6	29.7	36.6	18.6	**11.5**	98.1	52.3	67.8	31.5	**29.8**	906.3	462.3	936.0	399.7	**372.2**
			c_H	64.1	30.0	37.0	18.8	**10.2**	99.3	52.6	67.8	31.5	**22.4**	938.7	469.0	960.1	407.6	**321.2**
		d_L	c_L	67.3	31.0	36.5	18.6	**11.5**	103.6	56.1	67.8	30.9	**29.8**	1638.0	779.4	1664.9	624.6	**594.0**
			c_H	67.1	31.0	37.1	18.8	**9.1**	103.9	55.8	67.8	31.0	**20.9**	1598.6	760.2	1621.5	632.3	**431.8**
	180	d_S	c_L	69.3	31.5	37.9	19.3	**11.5**	108.0	56.1	67.8	31.3	**30.5**	900.3	448.6	813.5	350.3	**339.5**
			c_H	69.5	31.5	38.1	19.3	**10.5**	108.5	56.4	67.8	31.3	**23.5**	919.7	459.3	826.8	358.3	**299.4**
		d_L	c_L	71.3	32.9	38.6	19.6	**11.5**	111.8	59.9	67.8	**30.7**	30.8	1589.1	767.5	1453.6	**571.5**	571.9
			c_H	72.1	32.8	38.5	19.5	**9.5**	112.8	59.8	67.8	30.7	**22.0**	1584.8	757.3	1423.3	558.0	**418.0**
200	200	d_S	c_L	112.6	52.5	66.1	33.4	**20.3**	174.9	91.0	120.4	54.9	**53.4**	1679.4	846.4	1741.6	742.4	**711.9**
			c_H	111.2	51.8	65.5	33.1	**18.7**	172.1	89.8	120.4	56.2	**41.0**	1608.9	814.7	1702.1	747.6	**607.6**
		d_L	c_L	118.3	54.8	66.1	33.4	**20.3**	183.4	97.8	120.4	55.3	**53.4**	3022.9	1472.2	3103.0	1249.6	**1185.6**
			c_H	115.5	53.3	65.1	32.9	**17.2**	178.2	94.3	120.4	54.9	**39.2**	2873.5	1391.9	3049.8	1227.0	**924.9**
	240	d_S	c_L	120.8	55.1	67.7	34.0	**20.3**	188.8	97.6	120.4	54.2	**53.9**	1601.4	808.0	1489.1	627.5	**622.9**
			c_H	122.2	55.9	68.1	34.4	**18.9**	191.3	99.3	120.4	54.2	**42.3**	1628.3	826.7	1488.4	634.3	**563.6**
		d_L	c_L	125.0	56.9	68.0	34.2	**20.3**	195.4	102.6	120.4	54.7	**54.2**	2853.1	1391.4	2634.8	1063.9	**1049.2**
			c_H	123.7	56.5	67.4	34.1	**17.7**	192.8	101.7	120.4	55.0	**40.7**	2782.8	1362.5	2609.2	1060.9	**852.0**
Average				56.5	26.0	31.3	15.9	**8.9**	87.8	46.5	56.5	25.8	**21.9**	1045.4	510.9	1013.8	409.6	**355.5**

Table C.12: Complete numerical results determining the models' complexity for Category (A). All numbers indicate units of 10^3.