Technische Universität Dresden
Herausgeber: Der Rektor

Improved Flow-based Formulations for the Skiving Stock Problem

John Martinovic, Maxence Delorme, Manuel Iori, Guntram Scheithauer, Nico Strasdat

Preprint MATH-NM-01-2019

January 2019

Improved Flow-based Formulations for the Skiving Stock Problem

J. Martinovic[a,*], M. Delorme[b], M. Iori[c], G. Scheithauer[a], N. Strasdat[a]

[a]*Institute of Numerical Mathematics, Technische Universität Dresden, Germany*
[b]*School of Mathematics, The University of Edinburgh, United Kingdom*
[c]*DISMI, Università di Modena e Reggio Emilia, Italy*

Abstract

Thanks to the rapidly advancing development of (commercial) MILP software and hardware components, pseudo-polynomial formulations have been established as a powerful tool for solving cutting and packing problems in recent years. In this paper, we focus on the one-dimensional skiving stock problem (SSP), where a given inventory of small items has to be recomposed to obtain a maximal number of larger objects, each satisfying a minimum threshold length. In the literature, different modeling approaches for the SSP have been proposed, and the standard flow-based formulation has turned out to lead to the best trade-off between efficiency and solution time. However, especially for instances of practically meaningful sizes, the resulting models involve very large numbers of variables and constraints, so that appropriate reduction techniques are required to decrease the numerical efforts. For that reason, this paper introduces two improved flow-based formulations for the skiving stock problem that are able to cope with much larger problem sizes. By means of extensive experiments, these new models are shown to possess significantly fewer variables as well as an average better computational performance compared to the standard arcflow formulation.

Keywords: Cutting and Packing, Skiving Stock Problem, Arcflow Model, ILP

1. Introduction

Modeling frameworks based on graph theory have been investigated and successfully applied in different fields of cutting and packing [8, 12, 23, 26, 29, 35]. Probably, one of the earliest references is given by [39] where theoretical foundations on the relationship between certain *integer linear programs* (ILPs) and corresponding flow formulations have been proposed. However, at that time, computers were much slower than they are today, and addressing the exact solution of discrete optimization problems was not an easy task. More precisely, the computational times required to enumerate all the variables and constraints and to solve then the complete models exactly were too large. Hence, for a long time, research on these alternative formulations was rather theoretically motivated, so that, for instance, the numerical behavior and benefits of flow-based approaches could not be properly discussed in [39]; nevertheless, some properties suggesting computational advantages of such formulations are already presented therein.

Mainly after the publication of the famous book by Nemhauser and Wolsey [31] in 1988, also computational aspects (e.g., the strength of the considered models) became a central concern in ILP modeling. In recent years, the rapidly advancing development of (commercial) Mixed ILP (MILP) software and the steady progress in terms of powerful hardware components have successively contributed to the scientific importance of pseudo-polynomial formulations for the

[*]Corresponding author
Email addresses: john.martinovic@tu-dresden.de (J. Martinovic), maxence.delorme@ed.ac.uk (M. Delorme), manuel.iori@unimore.it (M. Iori), guntram.scheithauer@tu-dresden.de (G. Scheithauer), nico.strasdat@tu-dresden.de (N. Strasdat)

solution of cutting and packing problems [12, 30, 37]. In particular, a significant body of today's research is dealing with reduction techniques, as well as theoretical and algorithmic improvements [5, 9, 11] for the existing approaches.

In this paper, we want to promote this general idea by presenting two new and improved flow-based formulations for the one-dimensional *skiving stock problem* (or SSP for short). In its classical formulation, a threshold length $L \in \mathbb{N}$ and $m \in \mathbb{N}$ (different) item types that are specified by their length l_i and frequency (of occurence) b_i, $i \in I := \{1, \dots, m\}$, are considered. The SSP requires to recompose the given items in order to obtain a maximal number of objects, each having a length at least L, see also Fig. 1.

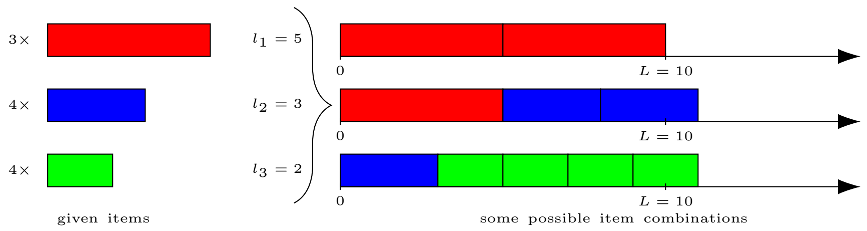

Figure 1: A simple schema of the skiving stock problem for the instance $E_0 = (m, l, L, b) = (3, (5, 3, 2), 10, (3, 4, 4))$.

In [4], this problem was introduced as the *dual bin packing problem* (DBPP) whenever $b_i = 1$ holds for all $i \in I$. The term *skiving stock problem* (SSP) has been proposed in the literature [40] for a particular practical application in the field of paper recycling [18]. Nowadays, both names are usually separated according to the typology presented in [38], meaning that

- the DBPP refers to highly heterogeneous instances with b_i, $i \in I$, close to (or equal to) one, see [22, 32], and
- the SSP describes instances where $\sum_{i \in I} b_i$ is large compared to the parameter m, see [26, 40].

The skiving stock problem plays an important role whenever an efficient and sustainable use of limited resources is intended. By way of example, such computations can be found in industrial production processes [18, 40], manufacturing and inventory plannings [2, 3], wireless communications [25, 34], or multiprocessor scheduling problems [1]. Further scientific relevance of the SSP is given by the fact that it may appear in holistic cutting-and-skiving scenarios [7, 18]. Observe that, although its connection to the extensively studied *cutting stock problem* (CSP) [12, 15, 19, 37] is obvious, both problems are not dual formulations in the sense of mathematical optimization.

Originally, the skiving stock problem was modeled by a pattern-based approach introduced in [40]. Although this model is known to possess a very tight LP relaxation [20, 27, 28], similiar to the CSP context [6, 21], the number of variables is exponential with respect to the number of items. Consequently, pseudo-polynomial models for the SSP (most notably the *arcflow model* and the *one-stick model*) have been established in the literature and were shown to always exhibit an equally good relaxation, as well as a reasonable computational performance for most of the considered instances [26]. However, especially for instances of practically meaningful sizes, the resulting models involve (very) large numbers of variables and constraints, so that appropriate reduction techniques are required to decrease the numerical efforts for solving the corresponding ILPs. For that reason, we introduce two improved graph-theoretic ILP formulations for the SSP that are based on the ideas of reversed loss arcs and reflected arcs [11], respectively. Both of these new approaches are tailored for addressing specific drawbacks of the existing flow-based modeling framework. More precisely, the first idea mainly eliminates superfluous sinks (and possibly also further redundant vertices) of the graph, whereas the main novelty of the second formulation is to only consider half of the bin capacity so that significantly reduced sets of vertices and arcs are obtained. As a consequence of these reductions, both approaches will be shown to lead to a better numerical behavior (on average) compared to the arcflow model from [26]. Moreover, dominance relations between the LP bounds obtained by the various models will be discussed. Altogether, it should be emphasized that our computational study also serves as a first systematic approach to collect and describe benchmarking instances (and their performance) for the skiving stock problem, which can form an experimental basis for future research articles. A preliminary version of this research, containing just one new model and very limited computational results, was presented at the International Conference on Operations Research 2018 (OR 2018, Brussels) as [24].

The paper is organized as follows: in the next section, we review the standard flow-based formulation for the SSP and discuss its most important reduction techniques. Afterwards, in Sect. 3, two improved modeling frameworks and related theoretical properties are presented. Then, the computational performance of the three models is compared in Sect. 4. Finally, we summarize the main ideas of this paper and give an outlook on future research.

2. The Arcflow Model

Throughout this paper we assume the item lengths to be ordered non-increasingly, i.e.,

$$L > l_1 > l_2 > \ldots > l_m > 0. \tag{1}$$

If required, we can easily obtain this order by a sorting algorithm, such as merge sort, with $\mathcal{O}(m \cdot \log m)$ operations.

Let $E = (m, l, L, b)$ be an instance of the SSP. Any combination of items whose lengths sum up to at least L is called *(packing) pattern* of E. More formally, a pattern can be referred to as an integer vector $a \in \mathbb{Z}_+^m$ where the ith component states how many items of type $i \in I$ are involved. Then, the pattern set is given by $P(E) = \left\{ a \in \mathbb{Z}_+^m \,\middle|\, l^\top a \geq L \right\}$. As this set (at least theoretically) contains an infinite number of elements, normally the restriction to *minimal patterns* (collected in the set $P^\star(E)$) is sufficient. Any minimal pattern possesses the property that none of its items can be removed without underrunning the threshold L.

Based on these preliminaries, we can define the quantity

$$\bar{v} := \max \left\{ l^\top a \,\middle|\, a \in P^\star(E) \right\},$$

which represents the largest total length of a minimal pattern.

Remark 1. *It is not necessary to know all the (exponentially many) minimal patterns to calculate $\bar{v}$. Instead, as we will see later in Algorithm 1, this value is "automatically" found while generating the arcflow graph.*

As a starting point, we consider the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ given by the set of vertices $\mathcal{V} = \{0, 1, \ldots, \bar{v}\}$ and the set of arcs

$$\mathcal{E} := \left\{ (p, q) \in \mathcal{V} \times \mathcal{V} \,|\, p < L,\ q - p \in \{l_1, \ldots, l_m\} \right\}.$$

In this setting, an arc $(p, q) \in \mathcal{E}$ represents the positioning[1] of an item of length $q - p = l_j \in \{l_1, \ldots, l_m\}$ with its left point at vertex $p \in \mathcal{V}$, see Fig. 2 for an example. Then, any path $s = (v_0, v_1, \ldots, v_k)$ in $\mathcal{G}$ from $v_0 = 0$ to some vertex $v_k \geq L$ corresponds to a pattern $a \in P(E)$. However, this basic approach still has some major drawbacks:

[1]Hence, the considered approach is sometimes referred to as a *position-indexed formulation*, see [26].

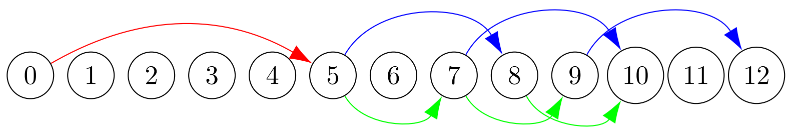

Figure 2: The vertex set and some arcs of $\mathcal{G}$ for the instance $E_0 = (3, (5, 3, 2), 10, (3, 4, 4))$. The colors of the item arcs are chosen according to Fig. 1. Altogether, $\mathcal{G}$ contains 13 vertices and 28 arcs.

- There are paths not corresponding to minimal patterns, such as the sequence $0 \to 5 \to 7 \to 9 \to 12$.
- There are redundant vertices that cannot be contained in paths starting at $v_0 = 0$ (e.g., $v = 1$).
- There are paths forming the same minimal pattern. For instance, the sequences $0 \to 5 \to 8 \to 10$ and $0 \to 5 \to 7 \to 10$ are both referring to the minimal pattern $a = (1, 1, 1)^\top$.

Consequently, reduction techniques are required to tackle these problems:

i) Obviously, only nonnegative integer linear combinations of the given item lengths have to be included in the set of vertices, since all other nodes cannot occur in a path $s = (v_0, v_1, \ldots, v_k)$ from $v_0 = 0$ to some $v_k \geq L$. Thus, the set $\mathcal{V}$ can be replaced by a reduced one (whose elements are mostly called *raster points*)

$$\mathcal{V}' := \left\{ v \in \mathcal{V} \,\middle|\, \exists\, a \in \mathbb{Z}_+^m : l^\top a = v \right\}.$$

This technique originates from the consideration of so-called *normal patterns* in early publications on two-dimensional cutting problems [10, 17], and refers to the observation [36, Criterion 3]. Even nowadays the theory of potential allocation points is a quite active field of research in cutting and packing [9, 14].

ii) In many cases, the previous reduction is not sufficient for vertices $v \geq L$, because the minimality of the corresponding item combinations is not ensured. By way of example, the instance $E = (2, (7, 2), 10, (7, 7))$ certainly leads to $12 \in \mathcal{V}'$, but there is no minimal pattern $a \in P^\star(E)$ with $l^\top a = 12$. For that reason, let

$$\mathcal{L} := \left\{ l^\top a \,\middle|\, a \in P^\star(E) \right\}$$

be the set of all possible total lengths of minimal patterns. Then, we can replace $\mathcal{V}'$ by

$$\mathcal{V}'' := \mathcal{V}' \cap (\{0, 1, \ldots, L-1\} \cup \mathcal{L}).$$

These observations also lead to a reduced set of arcs

$$\mathcal{E}' := \{(p, q) \in \mathcal{V}'' \times \mathcal{V}'' \,|\, p < L,\, q - p \in \{l_1, \ldots, l_m\}\}.$$

If we want to assign a path s in $\mathcal{G}' = (\mathcal{V}'', \mathcal{E}')$ to a minimal pattern $a \in P^\star(E)$, this identification is not necessarily unique, because $a \in \mathbb{Z}_+^m$ does not carry any information about the particular item order. For that reason, the consideration of monotonically decreasing paths (i.e., paths whose corresponding item lengths are sorted in non-increasing order) is proposed in the literature, see [33, Subsection 4.8.4.] or [35] for an exemplary application to the related CSP. Formally, this can be done by defining an index $\mu(p) \in I \cup \{0\}$ for all $p \in \mathcal{V}'' \setminus \mathcal{L}$ by

$$\mu(p) := \begin{cases} 0, & \text{if } p = 0, \\ \min\{i \in I \,|\, p - l_i \in \mathcal{V}'',\, i \geq \mu(p - l_i)\}, & \text{if } p \in \mathcal{V}'' \setminus (\mathcal{L} \cup \{0\}). \end{cases} \tag{2}$$

Remark 2. *Practically, this additional constraint can be incorporated by processing the item types in the order specified by assumption (1) within the graph generation, see also Algorithm 1.*

This leads to a new set of arcs

$$\mathcal{E}'' := \{(p,q) \in \mathcal{V}'' \times \mathcal{V}'' \mid p < L,\, q-p = l_i \in \{l_1,\dots,l_m\},\, i \geq \mu(p)\}$$

and a reduced graph $\mathcal{G}'' = (\mathcal{V}'', \mathcal{E}'')$, see Fig. 3 for an example.

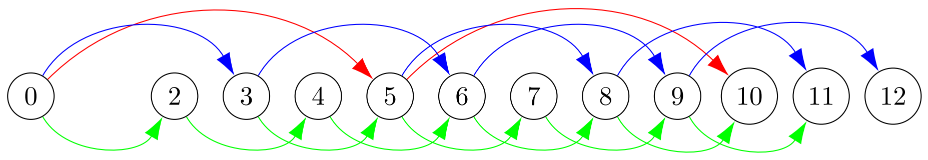

Figure 3: Arcflow graph $\mathcal{G}''$ for the instance $E_0 = (3,(5,3,2),10,(3,4,4))$. It contains 12 vertices and 17 arcs.

Remark 3. *The introduction of μ may not solve the problem of equivalent paths within $\mathcal{G}''$ entirely. Consider, by way of example, the instance E_0 where $\mathcal{G}''$ (illustrated in Fig. 3) contains the two paths*

$$0 \to 3 \to 5 \to 10 \quad \textit{and} \quad 0 \to 5 \to 8 \to 10$$

which both refer to the pattern $a = (1,1,1)^\top$. Moreover, the first path does not respect the monotonicity requirement. However, this problem cannot be avoided if, for instance, a (large) item length can be obtained by summing up two (or more) smaller item lengths.

Similar to [9, Algorithm 1], where a possible construction of the vertex set for the CSP scenario is described, the reduced arcflow graph can be generated in $\mathcal{O}(mL)$ by the procedure given in Algorithm 1.

Algorithm 1 `Create_arcflow_skiv`

```
1:  Input: $L$: Pattern threshold, $m$: Number of item types, $l$: Item lengths, $b$: Item availabilities
2:  $\mathcal{E}'' \leftarrow \emptyset$ ; $\mathcal{V}'' \leftarrow \emptyset$
3:  $M[0,\dots,L-1] \leftarrow 0$                    ▷ an array that keeps track of the possible tails
4:  $M[0] \leftarrow 1$                              ▷ node 0 is a possible tail
5:  for $i = 1$ to $m$ do
6:      for $k = L-1$ down to $0$ do
7:          if $M[k] = 1$ then                       ▷ if $k$ is a possible tail
8:              for $j = 1$ to $b_i$ do
9:                  if $k + (j-1) \times l_i \geq L$ then break     ▷ break if tail $\geq L$
10:                     $\mathcal{E}'' \leftarrow \mathcal{E}'' \cup \{(k+(j-1) \times l_i, k + j \times l_i)\}$
11:                     $\mathcal{V}'' \leftarrow \mathcal{V}'' \cup \{(k + j \times l_i)\}$
12:                 if $k + j \times l_i < L$ then
13:                     $M[k + j \times l_i] \leftarrow 1$
14:                 end if
15:             end for
16:         end if
17:     end for
18: end for
19: return $\mathcal{V}'', \mathcal{E}''$
```

In order to formulate the arcflow model in a preferably convenient way, we introduce the auxiliary index sets

$$A^-(q) := \{p \in \mathcal{V}'' \mid (p,q) \in \mathcal{E}''\}, \quad A^+(q) := \{r \in \mathcal{V}'' \mid (q,r) \in \mathcal{E}''\},$$

for every $q \in \mathcal{V}''$ and the set

$$E(i) := \{(p,q) \in \mathcal{E}'' \,|\, q - p = l_i\}$$

for every $i \in I$.

Remark 4. *Note that $A^-(q)$ and $A^+(q)$ model the predecessors and successors of vertex $q \in \mathcal{V}''$, respectively, whereas $E(i)$ collects all arcs referring to an item of length l_i. Moreover, the cases $A^-(q) = \emptyset$ and $A^+(q) = \emptyset$ can occur for some $q \in \mathcal{V}''$.*

Let $x_{pq} \in \mathbb{Z}_+$ denote the flow along arc $(p,q) \in \mathcal{E}''$, then we can state the

Arcflow Model of the Skiving Stock Problem

$$z_A = \sum_{q \in A^+(0)} x_{0q} \to \max$$

$$\text{s.t.} \quad \sum_{p \in A^-(q)} x_{pq} = \sum_{r \in A^+(q)} x_{qr}, \qquad q \in \mathcal{V}'' \setminus (\mathcal{L} \cup \{0\}), \tag{3}$$

$$\sum_{(p,q) \in E(i)} x_{pq} \leq b_i, \qquad i \in I, \tag{4}$$

$$x_{pq} \in \mathbb{Z}_+, \qquad (p,q) \in \mathcal{E}''. \tag{5}$$

Constraints (3) can be interpreted as a flow conservation: at every interior vertex, which is not a possible endpoint, the units of incoming flow (i.e., the number of items "terminating" at this position) have to equal the units of emanating flow (that is the number of items "starting" at this position). Thus, the objective function maximizes the total flow within $\mathcal{G}''$, that is, the total number of objects obtained. Conditions (4) ensure that the given item supply is not exceeded. In this arcflow model, the numbers of variables and constraints are $\mathcal{O}(mL)$ and $\mathcal{O}(m + L)$, respectively.

Although this model has turned out to be competitive in solving randomly generated instances [26], there are still some drawbacks connected to different aspects of this formulation:

(a) The size of the vertex set $\mathcal{V}''$ (and thus the size of the graph $\mathcal{G}''$) is strongly determined by the quantity $\bar{v}$, which is not known until the whole arcflow graph has been generated. Moreover, $\mathcal{G}''$ normally consists of multiple sinks (i.e., many different endpoints of patterns) whenever $\bar{v} > L$ holds.

(b) As indicated by Remark 3, in some cases, the graph still contains some symmetries or non-monotonous paths.

(c) The reduction caused by raster points is particularly successful at the beginning of the graph (i.e., in the first half of $\mathcal{V}$) whereas it tends to be nearly irrelevant in the second part.

Whereas the second problem may not be addressed in the general context (see Remark 3), we aim at tackling the remaining unfavorable properties by two new flow-based approaches in the following section.

3. Two Improved Flow-based Formulations

3.1. An Arcflow Model with Reversed Loss Arcs

As a starting point, we want to deal with problem (a) mentioned above. To this end, we assume that $L \in \mathcal{V}''$ holds without loss of generality. If this is not the case, then the given instance $E = (m, l, L, b)$ can be replaced by an equivalent instance $E' = (m, l, L', b)$ with $L' \geq L + 1$.

Now, instead of allowing multiple endpoints within the graph, we force any path to end at vertex $v = L$. This can be obtained by the following three steps:

(1) Any arc $(p, q) \in \mathcal{E}''$ with $q = L + r$ for some $r \geq 1$ is shifted back to $(p - r, L)$. After having completely executed this step, we let $\underline{v}$ denote the lowest-indexed vertex that is the tail of a shifted arc.

(2) Add reversed loss arcs (e, d) with $\underline{v} \leq d < e < L$ and $e = \text{succ}_{\mathcal{V}''}(d)$, where $\text{succ}_M(m)$ describes the successor of element $m \in M$ within the canonically ordered set $M \subseteq \mathbb{N}$.

(3) Now, any vertex that is not part anymore of at least one arc is deleted.

Let us refer to the resulting graph as $\mathcal{G}_L := (\mathcal{V}_L, \mathcal{E}_L)$, then principally the same optimization model as in Sect. 2 can be applied (now, of course, based on the updated sets $\mathcal{V}_L$ and $\mathcal{E}_L$), if the artificial loss arcs are also respected in the flow conservation conditions. We will briefly refer to this model as the *loss arcflow model* (or simply *larcflow*), and define by z_L its objective function.

Example 1. *The modified graph for the instance* $E_0 = (3, (5, 3, 2), 10, (3, 4, 4))$ *is depicted in Fig. 4.*

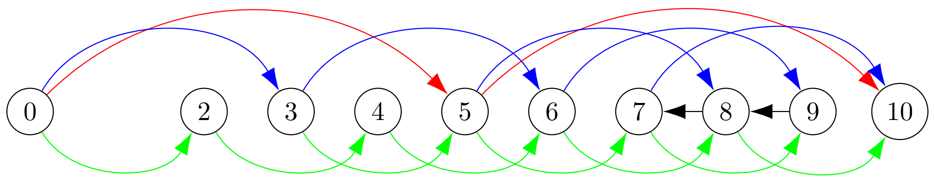

Figure 4: Arcflow graph $\mathcal{G}_L$ for the instance $E_0 = (3, (5, 3, 2), 10, (3, 4, 4))$. It contains 10 vertices and 17 arcs.

Here, the formerly existing arcs $(9, 12), (8, 11), (9, 11) \in \mathcal{E}''$ *have been shifted back to the arcs* $(7, 10), (7, 10), (8, 10) \in \mathcal{E}_L$*, and two additional reversed loss arcs have been introduced, namely* $(8, 7)$ *and* $(9, 8)$*. In this new setting, the vertices* 11 *and* 12 *do not appear anymore; and, by way of example, the path* $0 \to 3 \to 6 \to 9 \to 12$ *is now represented by* $0 \to 3 \to 6 \to 9 \overset{loss}{\to} 8 \overset{loss}{\to} 7 \to 10$.

Besides having eliminated the additional sinks of the graph (so that any path now terminates at $v = L$), there are some further advantageous, neutral and disadvantageous properties (marked by $(+)$, $(\pm)$ and $(-)$) that we want to mention here as a concluding summary of this subsection:

$(+)$ In our example, see Fig. 4, the new arcflow graph $\mathcal{G}_L$ contains the same number of arcs as before since the number of additional loss arcs is equal to the savings that are obtained by shifting arcs back (onto some already existing arcs). However, for practically meaningful problem sizes, this effect is not likely to happen so that, normally, a smaller number of variables can be expected (see also Sect. 4).

$(\pm)$ Typically, the loss arcflow model has (roughly) the same number of constraints as the standard arcflow formulation because the number of interior vertices is not likely to change considerably. However, in some (rather exceptional) cases, it may happen that interior vertices of $\mathcal{V}''$ can be deleted after having shifted the arcs according to Step (1) from above. On the other hand, for instances with sparse vertex set, it is also possible that the shifting of arcs can activate additional interior vertices. As we will see in our computational part (see Sect. 4), both of these phenomena do not play a very important role for larger instances, in general.

$(-)$ Possibly, the shifting of some arcs can produce additional symmetries or can destroy some monotonicity properties of the graph. For instance, the graph $\mathcal{G}_L$ depicted in Fig. 4 now contains a further possibility to represent the pattern $a = (1, 1, 1)^\top$. Moreover, at least from a theoretical point of view, arbitrarily long paths (patterns) can be obtained in $\mathcal{G}_L$ because the graph does not have to be acyclic anymore, see also Fig. 4.

3.2. The Reflect Arcflow Model

In this subsection, the problems (a) and (c) from the list presented in Sect. 2 shall be dealt with collectively. For that reason, we consider an approach basically introduced in [11] (for the CSP scenario), whose key idea is to reduce the set of vertices by decomposing any path into two subpaths (providing roughly half of the desired total length) that are linked by a reflected arc.

Example 2. *Let us again consider the instance* $E_0 = (3,(5,3,2),10,(3,4,4))$ *and the corresponding graph* $\mathcal{G}''$ *depicted in Fig. 3. By way of example, the description of the path* $0 \to 3 \to 6 \to 9 \to 12$ *actively uses five vertices and four different arcs. In a new setting, that only takes the vertices of the first half (of this path) into account, we could also describe this path as illustrated in Fig. 5.*

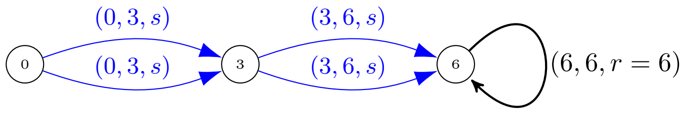

Figure 5: A preliminary schema to represent paths on only one half of the vertex set. The third index (s or r) of an arc is used to distinguish between standard arcs and reflected arcs, where in the latter case also the coordinate of the reflection point is mentioned as an auxiliary information.

Here, the two (identical) subpaths $(0,3,s) \to (3,6,s)$ *(each referring to two items of length* $l_2 = 3$*) are connected by a reflected arc* $(6,6,r=6)$ *(where the reflection point* $r = 6$ *is given by half of the previous total path length of* $l^\top a = 12$*). Hence, we obtain an equivalent description of the pattern* $a = (0,4,0)^\top$ *by only using three vertices and three different arcs.*

As skiving stock patterns can exhibit several lengths $L_t \in \mathcal{L} := \left\{ l^\top a \,\middle|\, a \in P^\star(E) \right\}$ (which are identical to the sinks of $\mathcal{G}''$), we usually would have to cope with multiple reflection points $r_t = L_t/2$ if the idea presented in [11] and the previous example is simply overtaken. Moreover, the elements of $\mathcal{L}$ are not known in advance and, hence, cannot be used to efficiently implement a reflect graph. Due to these difficulties, we propose another strategy to represent a pattern in the SSP context.

In order to avoid multiple reflection points, we again try to force any path (or pattern) to have exactly the length L by introducing appropriate loss arcs (but now in forward direction). Then, Algorithm 2 can be applied to build the reflect graph $\mathcal{G}_R := (\mathcal{U}, \mathcal{A})$ (for the SSP) with only one reflection point[2] $r := L/2$.

More precisely, we have that any arc $(u,v) \in \mathcal{E}''$ (appearing in Sec. 2)

- with $u < v \leq r$ is maintained as a *standard arc* $(u,v,s) \in \mathcal{A}$,
- with $u < r < v$ is transformed into a *reflected arc* $(u, L-v, r) \in \mathcal{A}$,
- with $r \leq u < v$ is deleted.

After having executed these steps, let $\underline{v}$ denote the lowest-indexed vertex that is the head of a reflected arc. Then, we add the *loss arcs* (d,e,l) with $d \in \mathcal{U}$, $\underline{v} \leq d < L/2$ and $e = \operatorname{succ}_{\mathcal{U}}(d)$ acting as the direct successor of d. Finally, we have to introduce a special reflected arc (not referring to any item length) $(L/2, L/2, r)$ to enable the combination of two subpaths providing exactly half of the bin capacity.

Example 3. *The reflect arcflow graph* $\mathcal{G}_R$ *for the instance* $E_0 = (3,(5,3,2),10,(3,4,4))$ *is depicted in Fig. 6. For the sake of a better comprehension, the colors of the arcs are now referring to the different arc types (standard, reflected and loss). Here, the path* $0 \to 3 \to 6 \to 9 \to 12$ *from*

[2] In order to avoid fractional coordinates it should be assumed that L is even. If required, this can be obtained by scaling all lengths of the instance by a factor 2. Note that this operation does not change the number of arcs.

Algorithm 2 `Create_reflect_skiv`

```
1:  Input: $L$: Pattern threshold, $m$: Number of item types, $l$: Item lengths, $b$: Item availabilities
2:  $\mathcal{A} \leftarrow \emptyset$ ; $\mathcal{U} \leftarrow \emptyset$
3:  $M[0,\dots,L/2] \leftarrow 0$                    ▷ an array that keeps track of the possible tails
4:  $M[0] \leftarrow 1$                               ▷ node 0 is a possible tail
5:  $\underline{v} = L/2$                             ▷ lowest-indexed vertex that is the head of a reflected arc
6:  for $i = 1$ to $m$ do
7:      $H[0\dots L/2] \leftarrow 0$                  ▷ an array that keeps track of the tails already processed
8:      for $j = 1$ to $b_i$ do
9:          for $k = L/2 - 1$ down to $0$ do
10:             if $H[k] = 0$ and $M[k] = 1$ then     ▷ if $k$ is unprocessed and a possible tail
11:                 $H[k] = 1$                        ▷ $k$ is now processed
12:                 if $k + l_i \le L/2$ then         ▷ if the arc is standard
13:                     $\mathcal{A} \leftarrow \mathcal{A} \cup \{(k, k + l_i, s)\}$
14:                     $\mathcal{U} \leftarrow \mathcal{U} \cup \{(k + l_i)\}$
15:                     $M[k + l_i] \leftarrow 1$
16:                 else                              ▷ if the arc is reflected
17:                     $\mathcal{A} \leftarrow \mathcal{A} \cup \{(k, L - (k + l_i), r)\}$
18:                     $\mathcal{U} \leftarrow \mathcal{U} \cup \{(L - (k + l_i))\}$
19:                     $\underline{v} \leftarrow \min\{\underline{v}, L - (k + l_i))\}$
20:                 end if
21:             end if
22:         end for
23:     end for
24: end for
25: $\mathcal{U} \leftarrow \mathcal{U} \cup \{L/2\}$
26: for $i \in \mathcal{U} : i \ge \underline{v}$ do $\mathcal{A} \leftarrow \mathcal{A} \cup \{(i, j, l)\colon\ j = \min(k \in \mathcal{U} : k > i)\}$    ▷ loss arcs
27: $\mathcal{A} \leftarrow \mathcal{A} \cup \{(L/2, L/2, r)\}$                ▷ allow paths to collide in $L/2$
28: return $\mathcal{U}, \mathcal{A}$
```

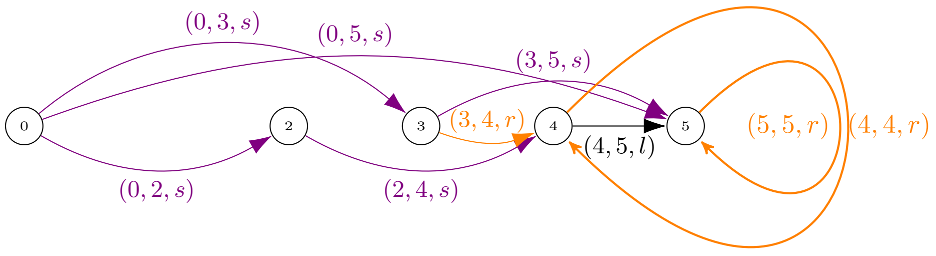

Figure 6: The reflect arcflow graph $\mathcal{G}_R$ for the instance $E_0 = (3, (5, 3, 2), 10, (3, 4, 4))$. It contains 5 vertices and 9 arcs.

the previous example is obtained by linking two (identical) reflected subpaths $0 \rightarrow 3 \overset{reflect}{\rightarrow} 4 \overset{loss}{\rightarrow} 5$ *by means of the reflected arc* $(5, 5, r)$. *Note that simply combining the two identical subpaths* $0 \rightarrow 3 \overset{reflect}{\rightarrow} 4$ *by means of the reflected arc* $(4, 4, r)$ *would correspond to the pattern* $a = (0, 4, 1)^\top$, *so that (at the moment) also non-minimal patterns can be discovered in the graph. However, this possibility will later be excluded by adapted flow conservation constraints.*

Observe that, even for this quite small example, the numbers of vertices and arcs could be reduced by roughly 50 percent compared to the standard arcflow model (from Fig. 3) or the model with reversed loss arcs (from Fig. 4). As mentioned in Sect. 2, these significant savings are obtained because the effect of normal patterns (or raster points) is noticed almost only at the beginning of the graph, whereas it tends to be irrelevant for the second half of the vertices (which is not modeled explicitly anymore).

Let us define $\mathcal{U}^\star := \mathcal{U} \setminus \{0\}$ and $\mathcal{A}^\star := \mathcal{A} \setminus \{(L/2, L/2, r)\}$, and let $\xi_{de\kappa}$ denote the flow along the generic arc $(d, e, \kappa) \in \mathcal{A}$ (where $\kappa \in \{s, r, l\}$ is possible); then the reflect arcflow model is given by

Reflect Arcflow Model of the Skiving Stock Problem

$$z_R = \sum_{(d,e,r)\in\mathcal{A}} \xi_{der} \to \max$$

$$\text{s.t.} \quad \sum_{(d,e,s)\in\mathcal{A}:\, e-d=l_i} \xi_{des} + \sum_{(d,e,r)\in\mathcal{A}:\, e=2r-d-l_i} \xi_{der} \le b_i, \qquad i \in I, \tag{6}$$

$$\sum_{(0,e,s)\in\mathcal{A}} \xi_{0es} + \sum_{(0,e,r)\in\mathcal{A}} \xi_{0er} = 2 \sum_{(d,e,r)\in\mathcal{A}} \xi_{der}, \tag{7}$$

$$\sum_{(d,e,s)\in\mathcal{A}} \xi_{des} + \sum_{(e,f,l)\in\mathcal{A}} \xi_{efl} = \dots$$

$$\dots \sum_{(d,e,r)\in\mathcal{A}} \xi_{der} + \sum_{(d,e,l)\in\mathcal{A}} \xi_{del} + \sum_{\kappa\in\{r,s\}} \sum_{(e,f,\kappa)\in\mathcal{A}} \xi_{ef\kappa}, \qquad e \in \mathcal{U}^\star, \tag{8}$$

$$\sum_{(d,e,l)\in\mathcal{A}} \xi_{del} + \sum_{(d,e,r)\in\mathcal{A}} \xi_{der} \ge \sum_{(e,f,l)\in\mathcal{A}} \xi_{efl}, \qquad e \in \mathcal{U}^\star, \tag{9}$$

$$\xi_{de\kappa} \in \mathbb{Z}_+, \qquad (d, e, \kappa) \in \mathcal{A}^\star, \tag{10}$$

$$\xi_{L/2,L/2,r} \in \mathbb{Z}. \tag{11}$$

The objective function maximizes the total flow on the reflected arcs. As (on balance) any unit of flow on a reflected arc is responsible for linking two subpaths to one single pattern, this quantity actually displays the number of obtained objects (with length $\ge L$). For this reason, we further have to demand that the total flow in the system is twice the objective value, see constraint (7). Moreover, conditions (6) require each item to be used at most b_i times, whereas constraints (8) model the (adapted) flow conservation at each interior vertex: note that (in a pattern sense) each flow entering a node through a standard arc has to be continued by a flow leaving the node through a standard or a reflected arc (classical flow conservation), or by a flow entering the node through a reflected arc (connection between two subpaths), including the additional flow brought and removed by the loss arcs.

Example 4. *The conditions explained so far would entail the following properties:*

i) Fortunately, the non-minimal pattern $a = (0, 4, 1)^\top$ from the previous example cannot be represented by a feasible path (consisting of two copies of $0 \to 3 \overset{reflect}{\to} 4$ linked by $(4, 4, r) \in \mathcal{A}$) in $\mathcal{G}_R$ anymore. To this end, let us consider the flow propagation for the arcs and vertices depicted in Fig. 7.

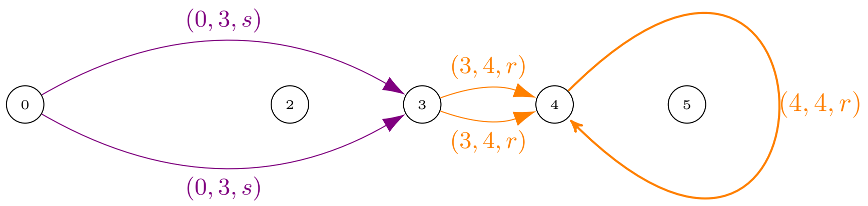

Figure 7: The former representation of the non-minimal pattern $a = (0, 4, 1)^\top$.

For the first interior node $e = 3$, we have two units of flow on the incoming standard arc $(0, 3, s)$ and two units of flow on the leaving reflected arc $(3, 4, r)$, so that condition (8) is satisfied. However, when looking at $e = 4$ there is no flow corresponding to the left-hand side of (8), whereas on the right-hand side we have two units of flow for the incoming reflected arc $(3, 4, r)$ and an additional nonnegative flow on the reflected arc $(4, 4, r)$. Hence, the flow conservation does not hold for the vertex $e = 4$, so that the non-minimal pattern cannot be part of a feasible flow (in $\mathcal{G}_R$) anymore.

ii) Contrary to the first point, at the moment, it would still be possible to link the two identical subpaths $0 \to 2 \to 4 \overset{loss}{\to} 5$ by means of the reflected arc $(5, 5, r)$ without violating constraints (8) for any of the involved interior vertices. As this would refer to a vector $a = (0, 0, 4)^\top$ (which does not represent a pattern) some further conditions are needed to limit the use of loss arcs.

Based on the second observation given in the previous example, loss arcs may only be used for special purposes. More precisely, they either have to directly follow another loss arc or a reflected arc, see (9). Then, by way of example, the vector $a = (0, 0, 4)^\top$ cannot be obtained anymore as a feasible connection of two subpaths.

One particularity of this model is that we have to allow negative flows on the (artificial) reflected arc $(L/2, L/2, r)$, see (11), to also enable two reflected paths to be merged together without violating the flow conservation and without miscounting the number of obtained patterns in the objective function.

Example 5. *As long as a nonnegative flow on the artificial reflected arc $(L/2, L/2, r) \in \mathcal{A}$ is demanded, the flow conservation conditions for $e = 5$ in the scenario depicted in Fig. 8 would lead to the same contradiction as in the previous example.*

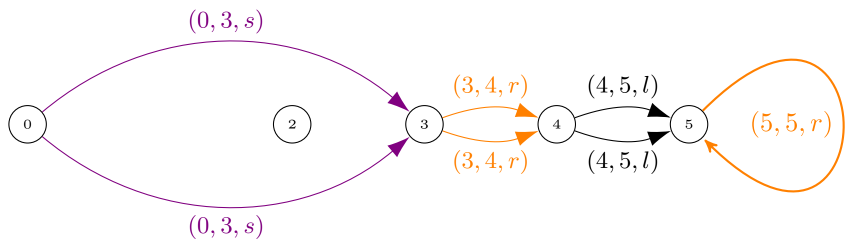

Figure 8: The pattern $a = (0, 4, 0)^\top$ is built by joining two subpaths with the help of the artificial reflected arc $(L/2, L/2, r) \in \mathcal{A}$.

Moreover, as already two units of flow are required on the reflected arc $(3, 4, r) \in \mathcal{A}$, this pattern would not be counted correctly (as one single pattern) in the objective function. To solve these two problems, the flow variable corresponding to $(L/2, L/2, r) \in \mathcal{A}$ is allowed to possess negative values. Then, with $\xi_{(0,3,s)} = \xi_{(3,4,r)} = \xi_{(4,5,l)} = 2$ and $\xi_{(5,5,r)} = -1$ all constraints are satisfied and the total flow on all the involved reflected arcs is equal to one, so that exactly one pattern is added to the objective value.

Example 6. *Another typical case where linking two reflected paths can sometimes be meaningful occurs if very large items may appear together in a single pattern, see Fig. 9 for an illustration of the exemplary instance $E_1 = (3, (18, 16, 8), 20, (10, 10, 10))$. Observe that, also in this setting, $\xi_{(L/2,L/2,r)}$ will become negative if, for instance, the two identical paths $0 \overset{reflect}{\to} 2 \overset{loss}{\to} 4 \overset{loss}{\to} 8 \overset{loss}{\to} 10$ shall be linked to obtain the pattern $a = (2, 0, 0)^\top$. For the sake of completeness, let us mention that an optimal solution (with $z_R^\star = 15$) is given by $\xi_{0,2,r} = \xi_{0,4,r} = \xi_{0,8,s} = \xi_{2,4,l} = \xi_{8,10,l} = 10$, $\xi_{4,8,l} = 20$, and $\xi_{10,10,r} = -5$.*

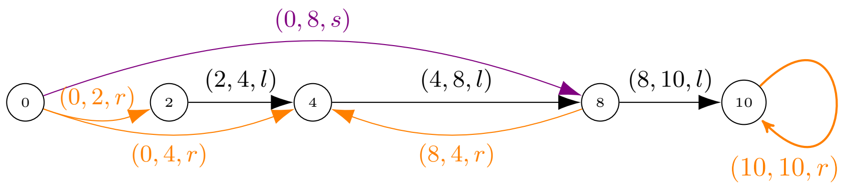

Figure 9: The reflect graph $\mathcal{G}_R$ for the instance $E_1 = (3, (18, 16, 8), 20, (10, 10, 10))$.

We also note that the reduction criteria proposed in [11, Proposition 5] stating that any feasible pattern can be represented by a pair of colliding paths whose reflected arc (d, e, r) has $d < e$ cannot be used for the SSP. Indeed, let us consider the pattern $a = (0, 0, 3)^\top$ from the previous example: it can only be obtained by the two colliding paths $0 \to 8 \overset{\text{reflect}}{\to} 4 \overset{\text{loss}}{\to} 8$ and $0 \to 8$, in which the reflected arc $(8, 4, r)$ has to be used.

In summary, modeling and generating the reflect arcflow formulation for the SSP is more challenging than in the cutting scenario [11] and requires (partly) independent techniques as well as considerable modifications of the corresponding ILP.

Altogether, this model possesses much fewer variables than the previous flow-based formulations (in Sect. 2 and Subsect. 3.1). As regards the number of constraints, no clear prediction can be made. On the one hand, our new approach allows for using much fewer flow conservation constraints (in most cases only roughly 50% of the original number) of type (8), whereas, on the other hand, an additional constraint (see (9)) has to be included for any vertex. Depending on the particular instance, these two effects may lead to slight differences (in both directions) if compared to the number of constraints of the standard arcflow model, see also Sect. 4.

3.3. Relations between the LP bounds

Whenever there are different modeling approaches for one and the same optimization problem, an important question deals with the strength of the bounds obtained by the respective 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. To this end, we will investigate the relationships between the optimal objective values of the three LP relaxations $z_A^{LP} := z_A^{LP}(E)$, $z_L^{LP} := z_L^{LP}(E)$, and $z_R^{LP} := z_R^{LP}(E)$ and discuss possible dominance relations.

A first important observation is given by:

Theorem 5. *For $E = (m, l, L, b) = (4, (8, 6, 4, 2), 12, (1, 1, 1, 1))$ we have $z_R^{LP} < z_A^{LP} < z_L^{LP}$.*

Proof. For the given instance, we have $z_R^{LP} = 1.5$, $z_A^{LP} = 1.6$, and $z_L^{LP} = 5/3$, obtained by the patterns (paths):

$$\begin{aligned}
R: &\quad 1 \times (1,0,1,0)^\top,\ \frac{1}{2} \times (0,2,0,0)^\top,\\
A: &\quad \frac{3}{5} \times (1,0,1,0)^\top,\ \frac{2}{5} \times (1,1,0,0)^\top,\ \frac{2}{5} \times (0,1,1,1)^\top,\ \frac{1}{5} \times (0,1,0,3)^\top,\\
L: &\quad \frac{2}{3} \times (1,0,1,0)^\top,\ \frac{1}{3} \times (1,0,0,2)^\top,\ \frac{1}{3} \times (0,2,0,0)^\top,\ \frac{1}{3} \times (0,1,1,1)^\top.
\end{aligned}$$

□

The main reason for the occuring differences is given by the fact that some patterns (paths) can be feasible in one graph, while they cannot be composed in another graph. In the example presented in the previous theorem, the patterns $a = (1,0,0,2)^\top$ and $a = (0,1,0,3)^\top$ cannot appear in the reflect graph because it is impossible to start a subpath with more than one copy of the item length $l_4 = 2$. On the other hand, the pattern $a = (0,2,0,0)^\top$ can for instance appear in the reflect framework, whereas it cannot be found in the standard arcflow formulation.

Given the information of Theorem 5, in the following result we clarify that there is no dominance relation between the reflect model on the one hand and either the standard arcflow model or the loss arcflow model on the other hand.

Theorem 6. *For $E = (m,l,L,b) = (2,(5,2),10,(1,5))$ we have $z_A^{LP} = z_L^{LP} < z_R^{LP}$.*

Proof. Indeed, for the given instance we have $z_R^{LP} = 1.5$ and $z_A^{LP} = z_L^{LP} = 1.4$, obtained by the patterns (paths):

$$R:\ 1 \times (0,5)^\top,\ \frac{1}{2} \times (2,0)^\top, \quad A\,\&\,L:\ 1 \times (1,3)^\top,\ \frac{2}{5} \times (0,5)^\top.$$

□

Again, the reason for the differing optimal values is based on different sets of feasible patterns (paths). For instance, the pattern $a = (2,0)^\top$ used in the solution of the reflect model does not appear in the other two graphs due to the construction presented in Algorithm 1, where the for-loop for $i = 1$ (with $1 \le j \le b_1$ implying $j = 1$) only generates one single arc (in the whole graph) corresponding to this item. Of course, also the reflect graph does only contain this arc once, but here two identical copies of this subpath ($0 \to 5$) can be linked to obtain a feasible pattern (that can then be used 0.5 times).

Based on the two instances given in the previous theorems, only the relationship between the optimal values z_A^{LP} and z_L^{LP} is not fully discussed yet. To answer this open question, we present the following concluding result.

Theorem 7. *For any instance $E = (m,l,L,b)$ we have $z_A^{LP} \le z_L^{LP}$.*

Proof. Consider a given instance $E = (m,l,L,b)$ and a corresponding feasible solution $x = (x_{pq})_{(p,q)\in\mathcal{E}''}$ of the LP relaxation of the standard arcflow model. Since the directed graph $\mathcal{G}''$ of that instance is acyclic, any flow (i.e., any feasible solution) decomposes into feasible paths $\gamma_1,\dots,\gamma_s$ from $v_0 = 0$ to some $v \ge L$ with, in general, fractional flow values $f_1,\dots,f_s \ge 0$ (being constant along the arcs of a fixed path). Now note that, by construction, any path γ in arcflow has a corresponding path $\widetilde{\gamma}$ in loss arcflow, where all the arcs except the last one(s) are the same:

- If the last arc (p,q) of γ ends after L, then loss arcs have to be added from the penultimate node p of γ to the node $L-(q-p)$ in loss arcflow. These loss arcs exist since loss arcs are created (between all active nodes) from $L-1$ to $L-\max\{l_i : i \in I\}$. The path $\widetilde{\gamma}$ is then completed by an arc $(L-(q-p), L)$ which exists as any arc ending after L in arcflow for item $i \in I$ is transposed to $(L-l_i, L)$ in loss arcflow.
- If the last arc of γ ends in L, then the paths γ and $\widetilde{\gamma}$ are exactly the same.

It is straightforward to show that these transformed paths $\widetilde{\gamma}_1,\dots,\widetilde{\gamma}_s$ (with the same flow values $f_1,\dots,f_s \ge 0$ as before) also satisfy the constraints of loss arcflow:

- As we did not touch arcs starting from $v_0 = 0$, the objective function (and also the objective value) remains the same.
- Any of the transformed paths $\widetilde{\gamma}_1,\dots,\widetilde{\gamma}_s$ satisfies the flow conservation constraints by construction; so the sum of all these paths will also satisfy it.

- The flow on the possibly new arcs of type $(L - l_i, L)$ in loss arcflow equals the flow on the arcs $(d, d + l_i)$ with $d + l_i \geq L$ in arcflow. Consequently, also the item limitations are obeyed.
- All flow variables are still nonnegative.

Altogether, we obtain a feasible solution $\widetilde{x}$ of the LP relaxation of loss arcflow with the same objective value. $\square$

Summarizing the main observations of this subsection, we can finally point out that arcflow dominates loss arcflow (in terms of the LP bound), whereas there are no further dominance relations among the three optimal objective values z_A^{LP}, z_L^{LP}, and z_R^{LP}.

4. Computational Experiments

In addition to the theoretical presentation that we provided for the new mathematical models of the SSP, their computational performance and average efficiency shall be discussed based on numerical experiments. For that purpose, three general categories of instances are considered:

(A) datasets designed for the SSP (or dual bin packing problem) from the literature,

(B) randomly generated instances,

(C) benchmarking instances originally introduced for the CSP (or for the bin packing problem)[3].

Based on the fact that the SSP is a relatively young and rather unknown optimization problem, only a few families of instances of the first type can be found in the relevant literature, whereas much more test sets are available for the CSP. As both problems share the same input data (but with a partly different meaning), we decided to also include these CSP benchmarks into our experimental study in order to provide a wider variety of data sets. It is not guaranteed that these typically challenging instances for the CSP will be difficult to solve in the SSP context as well; but at least some of them (as, for instance, the AI/ANI sets introduced in [12, Subsection 7.1.3]) will definitly preserve their hardness so that the consideration of CSP benchmarks is justified also from this point of view. Altogether, besides purely providing information on the practical behavior of our different models, this section also serves as a first systematic approach to collect and describe benchmarking instances for the SSP that can form an experimental basis for future research articles.

4.1. Data Sets

In what follows we aim at describing in more detail the three instance categories mentioned above.

(A) As regards datasets particularly designed for the SSP, the literature only offers a very few of them. Moreover, given the year (and the computational limits at that time) when they were published, some of these instances (e.g., those presented in [40, Table 3] with $L = 100$ and $m \leq 10$) nowadays have to be considered as too easy to notice any substantial differences between the various approaches. To the best of our knowledge, the only considerable systematic description of (larger) SSP benchmarking instances is contained in [32], where also some test sets previously appearing in [22] are proposed. Contrary to our definitions, these constructions mostly fix the total number $n = \sum_{i \in I} b_i$ (instead of the number m

[3]In fact, almost all of these instances have been proposed in the context of bin packing. However, since any bin packing problem can be equivalently formulated as a CSP, these benchmark sets will always be referred to as CSP instances in the following paragraphs.

of different item types) as an input parameter of a test class. Theoretically, given the fact that the numbers of variables and constraints (and, hence, the size of the ILP model) rather depend on m and L, this could lead to quite heterogeneous instances within one and the same test class. However, in our experiments we always have b_i, $i \in I$, equal to (or at least very close to) one so that $n \approx m$ holds, and this leads to instance sets possessing comparably complex ILP models for fixed input parameters n and L.

Unfortunately, the original instances from [32] are not available anymore, so we had to build our own instances based on the constructions described in that paper. More precisely, this first category consists of two test sets with the following input parameters:

(A1) These rather small instances were proposed in [22] and are described by $l_{\max} = 99$ and the following integers:

$$\begin{aligned} n &\in \{10, 20, 40, 60, 80, 100\}, \\ L &\in \{100, 120, 150, 200, 300, 400, 500\}, \\ l_{\min} &\in \{1, 20, 50\}. \end{aligned}$$

For any of these 126 combinations, we randomly generated 10 instances, each with uniformly distributed item lengths $l_i \in [l_{\min}, l_{\max}]$, $i \in I$. Note that the values of b_i do not have to be specified because any of the n items is drawn individually.

(A2) A second test set (that contains some larger instances) appearing in [32] is given by $l_{\max} = 999$ and the following integers:

$$\begin{aligned} n &\in \{60, 80, 100, 250, 500\}, \\ L &\in \{1000, 1200, 1500, 2000, 3000, 4000, 5000\}, \\ l_{\min} &\in \{1, 200, 500\}. \end{aligned}$$

For any of these 105 combinations, we randomly generated 10 instances, each with uniformly distributed item lengths $l_i \in [l_{\min}, l_{\max}]$, $i \in I$.

(B) Randomly generated instances for the SSP have previously been addressed in [26]. As all these problems were solved in reasonably short computation times, we intend to consider similar, but much larger instances in this second category. More precisely, any class is parametrized by a pair (m, L) in the following way:

$$\begin{aligned} m &\in \{200, 300, 400, 500\}, \\ L &\in \{10000, 20000, 30000, 50000\}. \end{aligned}$$

In accordance with [26], for any of the 16 combinations we randomly generated 10 instances, each with uniformly distributed item lengths $l_i \in [L/10, 3/4 \cdot L]$ and availabilities $b_i \in [1, 100]$.

(C) The third category of test sets consists of 1895 challenging instances that were originally invented for the CSP (or for the bin packing problem) over the past years. All these instances are well known benchmarks whose specifications are described in the relevant literature. More precisely, as similarly pinpointed in [11, Section 7.1], this category contains the three subclasses:

(C1) *Classical:* A set of 1615 instances presented in various articles in the last decades and having highly different characteristics. A complete description of these data sets can be found in [12].

(C2) *GI:* A total number of 80 instances (divided into four classes of 20 instances each) proposed in [16] partly involving extremely large threshold lengths $L \leq 1500000$.

(C3) *AI/ANI:* Two sets containing 100 challenging instances each that have been proposed in [12]. Here, the word "challenging" either means that already finding reasonable candidates for an optimal solution may be difficult (as in the AI case), or that (as regards the ANI instances) proving the optimality of a given solution becomes really hard since the additive integrality gap (measured with respect to the LP bound) is at least one.

Instances in set C have been gathered together in the BPPLIB [13] and can be found on the Internet by `http://or.dei.unibo.it/library/bpplib`. Instances from sets A and B are available from the authors upon request.

4.2. Methodology

All our experiments were executed on an AMD A10-5800K CPU with 16 GB RAM, using Gurobi 8.0.1 (imposed to run on a single thread) as MILP solver with a time limit of one hour. The model generation itself was performed in Python 3.6.4 and the Gurobi Python module. For any instance and for any formulation, we collected the following data:

- *opt*: an indicator stating whether the instance could be tackled successfully ($opt = 1$) or not ($opt = 0$) within the given time limit,
- $z^\star, z_c^\star$: optimal value of the ILP and the LP relaxation, respectively (if solvable),
- t, t_{LP}: total time (in seconds) needed to solve the ILP and the LP relaxation, respectively,
- n_{var}, n_{con}: number of variables and constraints appearing in the respective ILP,
- n_{nz}: number of nonzero entries in the constraint matrix of the respective ILP,
- LB, UB: best lower (upper) bound obtained for the optimal objective value.

Note that, given the large number of instances attempted, we will only refer to aggregate (in terms of *opt*) or averaged values for each considered class of instances instead of providing the individual results for any single example. Whenever an instance could not be solved to optimality, the time limit (one hour) will be used as the computation time t.

In addition to the comparison of the three (pure) ILP formulations, we also measure the effects of passing an initial feasible solution x_{heu} to the Gurobi solver. This feasible solution was always obtained by a heuristic proposed in [32] whose iterations can be described as follows: Choose the largest item type (say k) that is still available (i.e., with residual $b_k > 0$) and allocate $a_{kj} = \min\{\lfloor (L-1)/l_k \rfloor, b_k\}$ items of type k to the current bin j.

(I) If $a_{kj} < b_k$ holds, then we choose the smallest possible item type $i \in I$ with $b_i > 0$ so that increasing a_{ij} by one unit leads to a feasible pattern. (This step can always be performed by choosing $i = k$.)

(II) If $a_{kj} = b_k$ holds, then we continue with item type $k+1$ and try to assign as many items of that type to bin j without reaching the threshold L. Depending on the situation, then we have to apply step (I) or (II) with respect to the item index $k+1$.

This general procedure is repeated (while successively updating the values b_i, $i \in I$), until all items have been assigned or no feasible bin can be found anymore.

Based on the collected data, we will evaluate and compare the computational behavior of six solution strategies (three ILP formulations with or without the additional application of the heuristic) in this article. Note that, in [26], the original arcflow model turned out to provide the best computational performance for practically meaningful problem sizes (among the tested formulations) although it possesses slightly more variables and constraints compared to the onestick formulation, see [26, Table 6]. Consequently, the conventional arcflow formulation (introduced in Sect. 2) can be considered as the state-of-the-art solution strategy for skiving stock problems. Hence, our comparison also involves the most promising existing ILP formulation, so that a reasonable investigation of the various solution approaches is conducted.

4.3. Results and Discussion

In this subsection, we aim at presenting some of the results obtained by the computations described above. Note that, for the sake of simplicity, the arcflow model of Sect. 2, the loss arcflow model of Subsect. 3.1 and the reflect model of Subsect. 3.2 will mostly be referred to by their respective abbreviations "arcflow", "larcflow" and "reflect". Moreover, for the tables, we always use the short form "heu" instead of heuristic.

At first, we consider the instances of category (A) and state that the subset (A1) could be solved to optimality by any of the six variants within really short times, see Tab. 1. Hence, these instances are rather small to observe significant time differences among the considered formulations. However, having a look at especially the numbers of variables and nonzeros, we can notice a considerably less complex structure of the ILP model for reflect (having roughly a third of the variables and half of the nonzero matrix elements compared to arcflow) and larcflow. Moreover, slight differences for the various LP relaxation values could be observed underlining the (theoretical) fact that the considered formulations are not equivalent if continuous variables are considered.

Table 1: Computational results for the 1260 A1-instances: *opt* presents an aggregated number whereas all other data are average values

formulation		*opt*	t	t_{LP}	$z_c^\star$	n_{var}	n_{con}	n_{nz}
arcflow	with heu	1260	0.2	0.1	15.714466	4135.3	234.8	11431.8
	without heu	1260	0.5					
larcflow	with heu	1260	0.3	0.1	15.718284	3306.7	238.8	9774.7
	without heu	1260	0.5					
reflect	with heu	1260	0.1	0.0	15.720586	1493.5	193.8	5601.7
	without heu	1260	0.2					

To obtain a more precise picture of the six solution variants, we move forward to the harder instances contained in category (A2). At first, we summarize the averaged (or aggregated, in terms of *opt*) data in the same way as for the subset (A1), see Tab. 2. Here, we can see that both alternative formulations can solve more instances (out of a total number of 1050) than arcflow while also being faster on average (despite having a slightly worse LP bound). Again, this behavior is mainly due to significant savings related to the ILP model's complexity. Moreover, a very interesting observation is given by the usefulness of the heuristic for any of the three different approaches: both the number of instances solved to optimality and the average time needed to solve them become better if a starting point is passed to the ILP solver.

Table 2: Computational results for the 1050 A2-instances: *opt* presents an aggregated number whereas all other data are average values

formulation		*opt*	t	t_{LP}	$z_c^\star$	n_{var}	n_{con}	n_{nz}
arcflow	with heu	976	435.1	20.3	60.756419	197279.5	2206.3	543984.2
	without heu	946	626.3					
larcflow	with heu	982	415.6	19.4	60.758035	150606.8	2229.3	450638.8
	without heu	967	561.6					
reflect	with heu	1011	250.9	6.0	60.757427	61976.4	1820.9	235929.0
	without heu	983	391.8					

In Tab. 2 we are averaging over a wide variety of very heterogeneous instances. To highlight possible computational insights, we decided to perform more detailed analyses. To this end, Tab. 3–5 report on the values *opt* and t for one input parameter being fixed. Obviously, in almost all of the experiments and for almost any modeling approach, the application of the heuristic leads to better results with respect to the considered data. We also observe that the difficulty of the problem increases as: (i) the size of the smallest item decreases, (ii) the threshold length increases, and (iii) the number of items increases. At this position, we refer the interested reader to the appendix (Tab. A.15–A.17) to have a look at an even more detailed presentation of the results for the A2-instances solved by any of the six formulations.

Table 3: Averaged computation times and total numbers (in parentheses) of solved A2-instances for different values of l_{min}

	arcflow				larcflow				reflect			
l_{min}	with heu		without heu		with heu		without heu		with heu		without heu	
1	458.3	(322)	920.8	(295)	448.2	(324)	800.4	(310)	259.3	(338)	527.1	(316)
200	499.0	(318)	596.3	(316)	490.5	(324)	593.9	(319)	329.2	(331)	430.9	(328)
500	348.0	(336)	361.7	(335)	308.1	(334)	290.6	(338)	164.2	(342)	217.4	(339)

Table 4: Averaged computation times and total numbers (in parentheses) of solved A2-instances for different values of L

	arcflow				larcflow				reflect			
L	with heu		without heu		with heu		without heu		with heu		without heu	
1000	1.4	(150)	1.5	(150)	0.7	(150)	0.7	(150)	0.3	(150)	0.3	(150)
1200	3.8	(150)	3.8	(150)	6.6	(150)	6.5	(150)	8.0	(150)	5.4	(150)
1500	55.4	(150)	54.9	(150)	54.5	(150)	77.0	(150)	30.3	(150)	29.0	(150)
2000	303.7	(145)	303.5	(145)	302.3	(146)	259.6	(148)	107.5	(149)	148.8	(148)
3000	604.8	(138)	735.6	(136)	585.4	(139)	699.8	(135)	296.5	(143)	434.2	(141)
4000	903.6	(127)	1265.6	(121)	840.2	(130)	1140.4	(131)	584.7	(134)	904.1	(126)
5000	1173.1	(116)	2019.0	(94)	1119.6	(117)	1747.4	(103)	729.1	(135)	1220.6	(118)

Table 5: Averaged computation times and total numbers (in parentheses) of solved A2-instances for different values of n

	arcflow				larcflow				reflect			
n	with heu		without heu		with heu		without heu		with heu		without heu	
60	31.7	(210)	71.0	(210)	31.5	(210)	78.4	(210)	18.0	(210)	25.6	(210)
80	88.2	(209)	212.3	(209)	62.9	(209)	145.2	(209)	24.9	(210)	67.5	(210)
100	115.1	(209)	407.3	(199)	131.4	(209)	279.7	(207)	38.0	(210)	102.1	(209)
250	706.5	(189)	963.5	(183)	698.6	(189)	909.8	(185)	323.5	(202)	589.5	(192)
500	1234.1	(159)	1477.2	(145)	1153.7	(165)	1395.1	(156)	850.2	(179)	1174.1	(162)

In any of the experiments related to the subset (A2), the heuristic could be performed in much less than one second. Consequently, in addition to the advantageous effects observed earlier, the heuristic should always be applied to obtain a feasible starting point for the ILP solver. By way of example, Fig. 10 compares the solution times for the arcflow model with and without applying the heuristic. It can clearly be seen that there are many instances possessing roughly the same solution time in both cases, but there are also many instances whose solution time decreases drastically (partly from the time limit, indicating that there was no solution found previously) if the heuristic is applied prior to the optimization. Moreover, note that the opposite effect (i.e., a significant increase of the computation time) is also possible, but happens very rarely.

As a summary, we would like to further underline the positive effects of the heuristic by the data collected in Tab. 6–7. For any combination (F_1, F_2) of the six solution variants, we are counting the number of A2-instances:

- that are solved faster by formulation F_1 than by F_2, see Tab. 6. (Here, the word "faster" means that a time difference of at least 0.5 seconds could be observed.)
- that are solved by formulation F_1, but not by F_2, see Tab. 7.

Figure 10: Comparison of the ILP solution times for the A2-instances and the arcflow model: for each instance, a $\star$ is drawn at position (x, y), where the x- and y-values contain the times with and without heuristic, respectively. The red line displays the function $f(x) = x$.

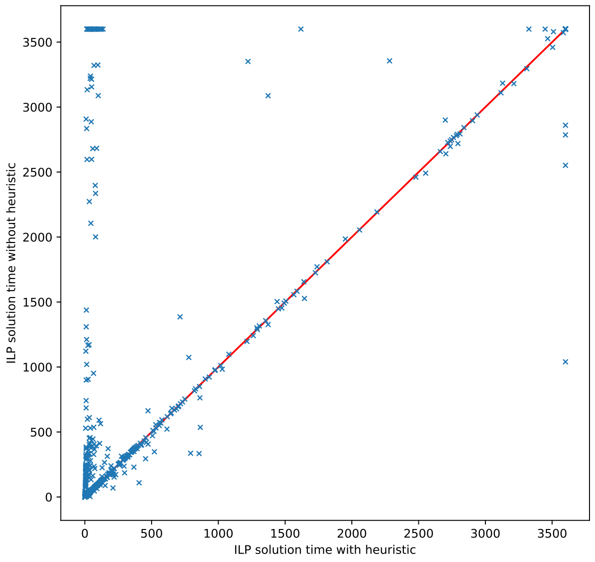

Table 6: Comparison between the solution times for the A2-instances: an entry x at position (F_1, F_2) gives the number of instances where formulation F_1 was faster than formulation F_2 (i.e., where a time difference of at least 0.5 seconds can be observed)

		arcflow		larcflow		reflect	
		with heu	without heu	with heu	without heu	with heu	without heu
arcflow	with heu	–	300	274	318	62	229
	without heu	154	–	159	201	48	74
larcflow	with heu	411	559	–	363	69	255
	without heu	380	510	208	–	70	109
reflect	with heu	738	766	660	660	–	410
	without heu	549	713	458	595	157	–

Table 7: Comparison between the A2-instances solved to optimality: an entry x at position (F_1, F_2) states the number of instances which were solved by formulation F_1 but not by F_2

		arcflow		larcflow		reflect	
		with heu	without heu	with heu	without heu	with heu	without heu
arcflow	with heu	–	34	18	38	6	22
	without heu	4	–	17	20	7	9
larcflow	with heu	24	53	–	45	7	27
	without heu	29	41	30	–	13	21
reflect	with heu	41	72	36	57	–	41
	without heu	29	46	28	37	13	–

In particular, if considering a fixed model type, than the entry at position (F_1, F_2) is always better than that at (F_2, F_1) if F_1 applies the heuristic and F_2 does not. Moreover, we can again state that the reflect formulation is superior to the other two approaches. This observation can also be verified by the data contained in Tab. 8. There, for any of the six formulations we counted the number of A2-instances where the respective approach possessed the fastest solution time. Again, being faster than another formulation is connected with a time difference of at least 0.5 seconds. Altogether, given the reasons presented (based on extensive computations) so far, for the remaining two categories of instances, (B) and (C), we will later only consider the three different models with the application of the heuristic.

One last thing we would like to report for the A2-instances deals with the strength of the respective LP relaxations. As to be clearly seen in Tab. 9, apart from the theoretically proved statement that arcflow dominates the loss arcflow model, any relation is possible among the other pairs of optimal values. This also emphasizes the fact that differences between the LP relaxation values do not only appear for the very simple instances particularly constructed for the proofs within the previous section, but also for larger instances of practically meaningful complexity.

Table 8: Number of A2-instances where the respective model was the fastest of the six solution approaches

formulation		#inst
arcflow	with heu	16
arcflow	without heu	3
larcflow	with heu	21
larcflow	without heu	28
reflect	with heu	378
reflect	without heu	117

Table 9: Comparison of the LP relaxation values of the three formulations: an entry x at position (F_1, F_2) gives the number of A2-instances with $z_c^\star(F_1) \leq z_c^\star(F_2)$ (measured with a tolerance of $\varepsilon = 10^{-8}$)

model	arcflow	larcflow	reflect
arcflow	1050	1050	1026
larcflow	951	1050	1004
reflect	963	1047	1050

Now, let us consider the instances from subset (B), see Tab. 10. Note that, although they are also randomly generated, they are completely different from the previous set (A) because the values b_i are usually much larger than one, so that typical SSP instances are built. Given the extensive discussion for the set (A) and the fact that randomly generated instances have already been partly adressed in the relevant literature [26], we only consider the average solution times and the numbers of solved instances per feasible combination of input data.
Contrary to the A-instances, the model with loss arcs is usually not better than the standard arcflow model, meaning that, in many cases, they possess a comparable performance, while in a few scenarios arcflow is also performing strictly better. A first explanation for this observation is given by the high degree of randomization within the Gurobi solution procedure. Hence, even if the same solver is called in both situations, different sub-routines or heuristics can appear in different orders, and this may influence the total solution efforts. On the other hand, even though the loss arcflow formulation basically has fewer variables (and roughly the same number of constraints), from our point of view, it does not counter balance the fact that its feasible integer solutions have a more complex structure that may be harder to deal with, compared to the standard flow propagation of arcflow, for a general ILP solver. Moreover, note that the B-instances are typically more complex and harder (on average) than those from the subset (A), and this especially leads to larger branching trees with much more nodes to visit. This is also due

Table 10: Results for the 160 B-instances (average solution times for the ILP and numbers of solved instances) always with application of heuristic

m	L	arcflow		larcflow		reflect	
200	10000	78.4	(10)	70.1	(10)	10.5	(10)
	20000	328.1	(10)	346.9	(10)	42.2	(10)
	30000	784.4	(10)	851.9	(10)	54.8	(10)
	50000	2812.5	(8)	2371.0	(9)	112.1	(10)
300	10000	156.8	(10)	148.7	(10)	27.7	(10)
	20000	477.7	(10)	630.0	(10)	117.9	(10)
	30000	1538.1	(10)	1696.8	(10)	137.0	(10)
	50000	3598.4	(1)	3581.0	(1)	319.0	(10)
400	10000	563.6	(9)	608.4	(9)	426.1	(10)
	20000	775.6	(10)	1008.3	(10)	540.8	(9)
	30000	2289.9	(9)	2464.1	(10)	910.9	(9)
	50000	3600.0	(0)	3600.0	(0)	1256.0	(8)
500	10000	1752.3	(8)	2989.0	(2)	1789.0	(6)
	20000	2854.9	(4)	2829.5	(5)	2921.9	(3)
	30000	3126.5	(4)	3600.0	(0)	3265.5	(1)
	50000	3600.0	(0)	3600.0	(0)	3600.0	(0)
avg/total		1771.1	(113)	1899.7	(106)	970.7	(126)

to the fact that the starting point provided by the heuristic leads to a lower bound whose absolute difference to the true optimal value[4] is much larger than in the previous settings. Altogether, under these conditions, finding better integer solutions becomes more and more important, so that the arcflow model can show advantages compared to the loss arcflow formulation. Similar argumentations can be applied to further explain the very few cases where the arcflow model is beating the reflect formulation: also here, a good integer solution found quickly can lead to significant improvements in dealing with the branch-and-bound tree. However, it should be stated that also for this kind of instances the reflect formulation is (on average) the best one, with significantly lower computation times and a larger total number of instances solved to optimality.

For the sake of completeness, as a last experiment we are also considering benchmark instances that have been presented for the related bin packing problem (or for the CSP) over the last couple of decades. Note that an instance which is hard in the bin packing case is not guaranteed to preserve its difficulty if interpreted as a dual bin packing instance, and vice versa. However, at least for some of the hard instances from the C-part, it is clear that they will remain hard in the SSP scenario. By way of example, let us particularly mention the ANI-instances, whose integrality gaps will also be at least one if interpreted as an SSP instance. The precise results can be found in Tab. 11. Note that, for some of the very large instances proposed by [16], we observed memory issues (i.e., the branching trees became too large), so that no data could be obtained. These situations are indicated by "–" in the table. Also here, the impression we got from the previous instance sets is again emphasized. For any subcategory (C1), (C2), and (C3), the reflect formulation is able to cope with the most instances, and possesses the lowest solution time. Again, the behavior between arcflow and loss arcflow is changing depending on the particular problem set: for instances where the main difficulty is the model size (e.g., Scholl 3, GI AA125 and GI BA125), loss arcflow seems better, while for instances instances where the

[4]Note that the optimal values for the B-instances are much higher than those from the A-part since $b_i \in [1, 100]$, $i \in I$, is allowed.

Table 11: Results for 1855 (out of 1895) C-instances (average solution times for the ILP and numbers of solved instances)

set	# inst	arcflow		larcflow		reflect	
Wäscher	17	83.0	(17)	117.6	(17)	257.3	(17)
Hard28	28	6.3	(28)	7.4	(28)	3.0	(28)
Falkenauer U	80	0.2	(80)	0.2	(80)	0.1	(80)
Falkenauer T	80	4.9	(80)	1.1	(80)	0.9	(80)
Schwerin 1	100	0.5	(100)	0.4	(100)	0.1	(100)
Schwerin 2	100	2.6	(100)	1.3	(100)	0.4	(100)
Scholl 1	720	0.1	(720)	0.1	(720)	0.0	(720)
Scholl 2	480	40.2	(480)	47.4	(480)	20.8	(480)
Scholl 3	10	3600.0	(0)	2306.6	(9)	65.1	(10)
avg/total (1)	1615	35.7	(1605)	30.0	(1614)	9.4	(1615)
GI AA125	20	3376.4	(3)	918.4	(20)	4.1	(20)
GI AB125	20	–	(–)	–	(–)	–	(–)
GI BA125	20	2964.6	(4)	1190.0	(19)	4.3	(20)
GI BB125	20	–	(–)	–	(–)	–	(–)
avg/total (2)	40	3170.5	(7)	1054.2	(39)	4.2	(40)
AI 201	50	99.2	(50)	117.5	(50)	26.3	(50)
AI 402	50	3219.5	(15)	3298.2	(10)	2016.9	(31)
ANI 201	50	1036.7	(43)	2843.5	(18)	238.5	(50)
ANI 402	50	3600.0	(0)	3600.0	(0)	3600.0	(0)
avg/total (3)	200	1988.9	(108)	2464.8	(78)	1470.4	(131)
avg/total (1-3)	1855	313.9	(1720)	314.6	(1731)	166.9	(1786)

main difficulty is to find a good integer solution (e.g., most of the Wäscher and AI 201 and 402) or to improve the upper bound (e.g., a few of the Wäscher and ANI 201 and 402), loss arcflow seems worse.

As a summary of all instance types, we would like to report about the average relations between the numbers of variables, constraints and nonzeros for the three models in Tab. 12. Both alternative formulations always possess much fewer variables and nonzeros compared to the standard arcflow model[5]. As regards the number of constraints, they are always roughly similar in any case, but we also observe that tiny differences (in both directions) compared to the standard arcflow model are possible. Altogether, note that in any of the three categories we observed instances that cannot be solved (in one hour and with the given computational environment) even by the most promising of the considered formulations. Consequently, our experiments do not only cover a significant set of instances, but also clearly point out the limits of what can currently be solved within a reasonable amount of time. We better highlight this fact in Tab. 13, where we show the behavior of the reflect formulation for the very hard instances of the categories (C2) and (C3).

Finally, we would like to refer the interested reader to Tab. A.14 in our appendix, where the quality of the heuristic has been evaluated also for the sets (B) and (C).

[5]Especially for the reflect formulation, the savings go up to about 90% (on average) as observed for the number of variables in the B-part. We would like to stress again that these B-instances were the only pure SSP instances, whereas the other ones possess very small values of b_i, so that they should rather be termed as DBPP instances.

Table 12: Relative numbers of variables, constraints and nonzeros averaged over all instances of the respective category (where the value of the arcflow model is normalized to 1.00)

formulation	A1			A2			B			C		
	n_{var}	n_{con}	n_{nz}	n_{var}	n_{con}	n_{nz}	n_{var}	n_{con}	n_{nz}	n_{var}	n_{con}	n_{nz}
arcflow	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
larcflow	0.73	1.06	0.78	0.60	1.05	0.66	0.52	1.00	0.62	0.65	1.02	0.71
reflect	0.34	0.96	0.51	0.23	1.00	0.34	0.11	1.02	0.20	0.27	0.95	0.42

Table 13: Some further results for very hard CSP-bechmark instances also showing the limits of the reflect formulation

set	#inst	arcflow	larcflow	reflect
GI AA250	20	– (–)	– (–)	103.9 (20)
GI AA500	20	– (–)	– (–)	3600.0 (0)
GI BA250	20	– (–)	– (–)	112.2 (20)
GI BA500	20	– (–)	– (–)	3536.7 (2)
AI 600	50	– (–)	3600 (0)	3588.2 (1)

5. Conclusions

In this paper, we investigated an existing and two new arcflow formulations for the skiving stock problem. The main idea of the two new approaches is given by avoiding superfluous nodes that are larger than the capacity L or by only considering the first half of the vertex set and to model a pattern as the sum of two subpaths that are connected by a reflected arc. For these three formulations, the quality of the continuous relaxation has been investigated from a theoretical point of view. Moreover, based on extensive numerical tests with a wide variety of differently characterized instances, we have shown that especially the new reflect model possesses significantly fewer variables and much better solution times (thus also resulting to a larger number of instances solved to optimality) compared to the original formulation. Consequently, the reflect arcflow model may be seen as a powerful tool for solving (large) instances of the SSP in reasonably short time. As future research, we envisage the use of these enhanced arcflow models to other relevant cutting and packing problems.

References

[1] Alvim, A.C.F., Ribeiro, C.C., Glover, F., Aloise, D.J.: A hybrid improvement heuristic for the one-dimensional bin packing problem. Journal of Heuristics 10(2), 205–229 (2004)

[2] Arbib, C., Di Iorio, C.F., Marinelli, F., Rossi, F.: Cutting and Reuse: an application from automobile component manufacturing. Operations Research 50(6), 923–934 (2012)

[3] Arbib, C., Marinelli, F.: Integrating process optimization and inventory planning in cutting-stock with skiving option: An optimization model and its application. European Journal of Operational Research 163(3), 617–630 (2005)

[4] Assmann, S.F., Johnson, D.S., Kleitman, D.J., Leung, J.Y.-T.: On a dual version of the one-dimensional Bin Packing Problem. Journal of Algorithms 5, 502–525 (1984)

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

[6] Caprara, A., Dell'Amico, M., Díaz Díaz, J.C., Iori, M., Rizzi, R.: Friendly Bin Packing Instances without Integer Round-up Property. Mathematical Programming-B 150, 5–17 (2015)

[7] Chen, Y., Song, X., Ouelhadj, D., Cui, Y.: A heuristic for the skiving and cutting stock problem in paper and plastic film industries. International Transactions in Operational Research 26(1), 157-179 (2019)

[8] Clautiaux, F., Sadykov, R., Vanderbeck, F., Viaud, Q.: Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem. Discrete Optimization 29, 18–44 (2018)

[9] Côté, J.-F., Iori, M.: The Meet-in-the-Middle Principle for Cutting and Packing Problems. INFORMS Journal on Computing 30(4), 625–786 (2018)

[10] Christofides, N., Whitlock, C.: An Algorithm for Two-Dimensional Cutting Problems. Operations Research 25(1), 30–44 (1977)

[11] Delorme, M., Iori, M.: Enhanced Pseudo-Polynomial Formulations for Bin Packing and Cutting Stock Problems. *to appear in:* INFORMS Journal on Computing, *currently available online as a Technical Report:* `http://www.optimization-online.org/DB_HTML/2017/10/6270.html` (2019)

[12] 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)

[13] Delorme, M., Iori, M., Martello, S.: BPPLIB: a library for bin packing and cutting stock problems. Optimization Letters 12(2), 235–250 (2018)

[14] Fischer, A., Scheithauer, G.: Cutting and Packing Problems with Placement Constraints. *In:* G. Fasano and J.D. Pinter (eds.): Optimized Packings with Applications, 119–156, Springer (2015)

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

[16] Gschwind, T., Irnich, S.: Dual inequalities for stabilized column generation revisited. INFORMS Journal on Computing (28), 175–194 (2016)

[17] Herz, J.C.: Recursive Computational Procedure for Two-dimensional Stock Cutting. IBM Journal of Research and Development 16(5), 462–469 (1972)

[18] Johnson, M.P., Rennick, C., Zak, E.J.: Skiving addition to the cutting stock problem in the paper industry. SIAM Review 39(3), 472–483 (1997)

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

[20] Kartak, V., Ripatti, A.: Large proper gaps in bin packing and dual bin packing problems. *to appear in:* Journal of Global Optimization, *available online:* `https://link.springer.com/article/10.1007/s10898-018-0696-0`, (2018)

[21] Kartak, V., Ripatti, A., Scheithauer, G., Kurz, S.: Minimal proper non-IRUP instances of the one-dimensional cutting stock problem. Discrete Applied Mathematics 187, 120–129 (2015)

[22] Labbé, M., Laporte, G., Martello, S.: An exact algorithm for the dual bin packing problem. Operations Research Letters 17, 9–18 (1995)

[23] Macedo, R., Alves, C., Valério de Carvalho, J.M.: Arc-Flow Model for the Two-Dimensional Guillotine Cutting Stock Problem. Computers & Operations Research 37, 991–1001 (2010)

[24] Martinovic, J., Delorme, M., Iori, M., Scheithauer, G.: An Improved Arcflow Model for the Skiving Stock Problem. *to appear in:* Operations Research Proceedings 2018 (2019)

[25] Martinovic, J., Jorswieck, E., Scheithauer, G., Fischer, A.: Integer Linear Programming Formulations for Cognitive Radio Resource Allocation. IEEE Wireless Communication Letters 6(4), 494–497 (2017)

[26] Martinovic, J., Scheithauer, G.: Integer linear programming models for the skiving stock problem. European Journal of Operational Research 251(2), 356–368 (2016)

[27] Martinovic, J., Scheithauer, G.: Integer rounding and modified integer rounding for the skiving stock problem. Discrete Optimization 21, 118–130 (2016)

[28] Martinovic, J., Scheithauer, G.: An Upper Bound of $\Delta(E) < 3/2$ for Skiving Stock Instances of the Divisible Case. Discrete Applied Mathematics 229, 161–167 (2017)

[29] Martinovic, J., Scheithauer, G.: The skiving stock problem and its relation to hypergraph matchings. Discrete Optimization 29, 77–102 (2018)

[30] Martinovic, J., Scheithauer, G., de Carvalho, V.: 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)

[31] Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. Wiley, New York (1988)

[32] Peeters, M., Degraeve, Z.: Branch-and-price algorithms for the dual bin packing and maximum cardinality bin packing problem. European Journal of Operational Research 170(2), 416–439 (2006)

[33] 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)

[34] Tragos, E.Z., Zeadally, S., Fragkiadakis, A.G. Siris, V.A.: Spectrum Assignment in Cognitive Radio Networks: A Comprehensive Survey. IEEE Communications Surveys & Tutorials 15(3), 1108 – 1135 (2013)

[35] Valério de Carvalho, J.M.: Exact Solution of Cutting Stock Problems Using Column Generation and Branch-and-Bound. International Transactions in Operational Research 5, 35–44 (1998)

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

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

[38] Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research 183, 1109–1130 (2007)

[39] Wolsey, L.A.: Valid Inequalities, Covering Problems and Discrete Dynamic Programs. Annals of Discrete Mathematics 1, 527–538 (1977)

[40] Zak, E.J.: The skiving stock problem as a counterpart of the cutting stock problem. International Transactions in Operational Research 10, 637–650 (2003)

Appendix. Additional Computational Results

Table A.14: Relative gap $\Delta_r = (z^\star - z_{heu})/z^\star$ between heuristic value and optimal value for all instance categories. If optimality was not proved, $z^\star \approx UB$ with the upper bound UB (from the reflect model) found by Gurobi was taken instead.

interval	#inst (A1)	#inst (A2)	#inst (B)	#inst (C)
$\Delta_r = 0$	905	398	0	628
$\Delta_r \in (0, 0.01]$	0	18	0	45
$\Delta_r \in (0.01, 0.02]$	0	41	0	112
$\Delta_r \in (0.02, 0.05]$	99	180	2	453
$\Delta_r \in (0.05, 0.10]$	173	289	48	428
$\Delta_r \in (0.10, 0.20]$	62	124	110	182
$\Delta_r > 0.20$	21	0	0	7
	1260	1050	160	1855

Table A.15: Arcflow Results for A2-instances (average solution times for the ILP and number of solved instances)

		with heuristic					without heuristic				
L	l_{min}/n	60	80	100	250	500	60	80	100	250	500
1000	1	0.5 (10)	0.6 (10)	1.0 (10)	3.6 (10)	10.5 (10)	0.5 (10)	0.7 (10)	1.1 (10)	4.0 (10)	10.4 (10)
	200	0.1 (10)	0.1 (10)	0.2 (10)	0.8 (10)	2.1 (10)	0.1 (10)	0.2 (10)	0.2 (10)	1.1 (10)	2.9 (10)
	500	0.0 (10)	0.0 (10)	0.0 (10)	0.2 (10)	0.8 (10)	0.0 (10)	0.0 (10)	0.1 (10)	0.3 (10)	1.1 (10)
1200	1	0.9 (10)	1.1 (10)	1.9 (10)	10.4 (10)	26.7 (10)	0.8 (10)	1.1 (10)	1.9 (10)	10.2 (10)	26.3 (10)
	200	0.2 (10)	0.4 (10)	0.5 (10)	3.3 (10)	9.9 (10)	0.2 (10)	0.4 (10)	0.6 (10)	3.3 (10)	10.3 (10)
	500	0.0 (10)	0.1 (10)	0.1 (10)	0.3 (10)	1.0 (10)	0.0 (10)	0.1 (10)	0.1 (10)	0.5 (10)	1.4 (10)
1500	1	5.7 (10)	7.7 (10)	15.7 (10)	203.3 (10)	534.1 (10)	5.0 (10)	5.9 (10)	15.6 (10)	199.2 (10)	533.7 (10)
	200	0.6 (10)	1.1 (10)	2.1 (10)	7.9 (10)	46.8 (10)	0.6 (10)	1.0 (10)	2.1 (10)	7.8 (10)	46.5 (10)
	500	0.1 (10)	0.2 (10)	0.3 (10)	1.4 (10)	3.9 (10)	0.1 (10)	0.2 (10)	0.3 (10)	1.3 (10)	3.8 (10)
2000	1	17.9 (10)	41.3 (10)	72.9 (10)	510.4 (10)	2975.2 (5)	24.2 (10)	47.1 (10)	58.6 (10)	463.1 (10)	2970.0 (5)
	200	5.7 (10)	9.9 (10)	23.7 (10)	181.4 (10)	670.9 (10)	5.9 (10)	9.8 (10)	23.6 (10)	181.7 (10)	671.7 (10)
	500	0.5 (10)	0.9 (10)	2.3 (10)	17.1 (10)	24.7 (10)	0.7 (10)	2.4 (10)	3.5 (10)	25.2 (10)	64.9 (10)
3000	1	39.2 (10)	55.5 (10)	206.9 (10)	1471.1 (9)	2580.7 (5)	112.2 (10)	177.1 (10)	342.1 (10)	2225.1 (10)	3489.0 (2)
	200	11.9 (10)	84.3 (10)	240.2 (10)	821.6 (10)	3056.0 (4)	31.0 (10)	83.7 (10)	205.9 (10)	828.5 (10)	3043.8 (4)
	500	7.4 (10)	19.5 (10)	30.8 (10)	106.1 (10)	341.3 (10)	7.3 (10)	19.3 (10)	30.7 (10)	105.8 (10)	332.5 (10)
4000	1	46.1 (10)	22.1 (10)	166.5 (10)	1382.0 (7)	1836.2 (5)	231.2 (10)	478.0 (10)	1453.2 (9)	3185.5 (4)	3572.4 (1)
	200	81.1 (10)	175.0 (10)	281.7 (10)	2053.1 (8)	3600.0 (0)	130.7 (10)	233.0 (10)	342.7 (10)	1863.9 (10)	3600.0 (0)
	500	66.7 (10)	144.1 (10)	228.1 (10)	1413.6 (9)	2058.2 (8)	72.0 (10)	144.8 (10)	201.2 (10)	1415.7 (9)	2059.7 (8)
5000	1	54.9 (10)	677.6 (9)	60.8 (10)	1841.4 (5)	1158.7 (7)	349.6 (10)	1883.9 (9)	3361.0 (3)	3388.0 (2)	3600.0 (0)
	200	163.0 (10)	259.3 (10)	509.9 (9)	1559.2 (7)	3600.0 (0)	306.9 (10)	827.1 (10)	1695.5 (7)	3107.3 (5)	3600.0 (0)
	500	162.3 (10)	351.8 (10)	571.7 (10)	3247.5 (4)	3377.6 (5)	211.6 (10)	541.6 (10)	814.2 (10)	3216.6 (3)	3381.6 (5)

Table A.16: Loss-Arcflow Results for A2-instances (average solution times for the ILP and number of solved instances)

		with heuristic					without heuristic				
L	l_{min}/n	60	80	100	250	500	60	80	100	250	500
1000	1	0.5 (10)	0.7 (10)	0.9 (10)	2.4 (10)	4.7 (10)	0.6 (10)	0.7 (10)	0.9 (10)	2.2 (10)	4.2 (10)
	200	0.1 (10)	0.1 (10)	0.1 (10)	0.5 (10)	1.0 (10)	0.1 (10)	0.1 (10)	0.1 (10)	0.5 (10)	1.0 (10)
	500	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)
1200	1	1.0 (10)	1.8 (10)	2.2 (10)	8.7 (10)	79.6 (10)	1.0 (10)	1.7 (10)	2.2 (10)	7.7 (10)	80.2 (10)
	200	0.1 (10)	0.3 (10)	0.4 (10)	1.3 (10)	2.7 (10)	0.2 (10)	0.3 (10)	0.4 (10)	1.2 (10)	2.6 (10)
	500	0.0 (10)	0.0 (10)	0.0 (10)	0.1 (10)	0.1 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.1 (10)	0.2 (10)
1500	1	2.1 (10)	19.0 (10)	9.8 (10)	213.5 (10)	545.1 (10)	2.1 (10)	9.1 (10)	12.4 (10)	189.3 (10)	895.1 (10)
	200	0.5 (10)	1.0 (10)	1.3 (10)	5.9 (10)	17.1 (10)	0.4 (10)	1.0 (10)	1.4 (10)	8.2 (10)	34.9 (10)
	500	0.1 (10)	0.1 (10)	0.1 (10)	0.4 (10)	1.2 (10)	0.0 (10)	0.1 (10)	0.1 (10)	0.4 (10)	1.0 (10)
2000	1	9.8 (10)	31.0 (10)	89.1 (10)	445.9 (10)	2876.3 (6)	30.3 (10)	80.4 (10)	60.5 (10)	657.2 (10)	2241.3 (8)
	200	2.4 (10)	7.9 (10)	20.7 (10)	156.9 (10)	887.8 (10)	2.3 (10)	12.8 (10)	17.7 (10)	143.0 (10)	640.7 (10)
	500	0.2 (10)	0.2 (10)	0.4 (10)	1.7 (10)	3.9 (10)	0.2 (10)	0.3 (10)	0.5 (10)	1.9 (10)	4.6 (10)
3000	1	40.1 (10)	43.0 (10)	518.6 (9)	1352.0 (9)	2512.1 (5)	115.5 (10)	184.2 (10)	335.5 (10)	1727.2 (9)	3235.0 (3)
	200	16.5 (10)	77.8 (10)	178.0 (10)	1206.2 (10)	2418.5 (6)	67.3 (10)	147.4 (10)	197.5 (10)	1271.8 (9)	2890.3 (4)
	500	3.7 (10)	8.1 (10)	12.0 (10)	160.6 (10)	234.2 (10)	3.8 (10)	7.9 (10)	12.0 (10)	69.7 (10)	232.6 (10)
4000	1	56.2 (10)	26.4 (10)	144.1 (10)	1257.7 (8)	1786.0 (6)	224.2 (10)	405.7 (10)	781.8 (10)	2699.0 (8)	3297.1 (6)
	200	101.3 (10)	204.9 (10)	369.2 (10)	2199.5 (6)	3195.7 (3)	195.7 (10)	267.8 (10)	601.1 (10)	2231.5 (7)	3373.8 (2)
	500	64.1 (10)	106.0 (10)	96.4 (10)	1366.7 (9)	1628.9 (8)	58.4 (10)	99.0 (10)	124.6 (10)	950.6 (10)	1795.5 (8)
5000	1	65.0 (10)	459.0 (9)	77.3 (10)	1845.4 (5)	1161.8 (7)	386.9 (10)	1036.3 (9)	2108.0 (7)	3600.0 (0)	3600.0 (0)
	200	129.7 (10)	276.9 (10)	521.9 (10)	1561.8 (9)	3600.0 (0)	378.5 (10)	570.9 (10)	1088.4 (10)	3091.1 (6)	3545.2 (1)
	500	167.9 (10)	57.0 (10)	717.6 (10)	2883.3 (3)	3270.0 (4)	179.6 (10)	223.5 (10)	527.9 (10)	2452.3 (6)	3422.7 (4)

Table A.17: Reflect Results for A2-instances (average solution times for the ILP and number of solved instances)

		with heuristic					without heuristic				
L	l_{min}/n	60	80	100	250	500	60	80	100	250	500
1000	1	0.1 (10)	0.2 (10)	0.3 (10)	1.0 (10)	1.9 (10)	0.2 (10)	0.2 (10)	0.3 (10)	1.1 (10)	1.8 (10)
	200	0.0 (10)	0.0 (10)	0.0 (10)	0.1 (10)	0.4 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.2 (10)	0.4 (10)
	500	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)
1200	1	0.4 (10)	0.4 (10)	1.0 (10)	8.1 (10)	108.8 (10)	0.3 (10)	0.4 (10)	1.0 (10)	10.1 (10)	67.1 (10)
	200	0.0 (10)	0.1 (10)	0.1 (10)	0.4 (10)	1.1 (10)	0.0 (10)	0.1 (10)	0.1 (10)	0.5 (10)	1.4 (10)
	500	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.1 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.0 (10)	0.1 (10)
1500	1	0.6 (10)	1.4 (10)	2.8 (10)	63.6 (10)	338.5 (10)	0.8 (10)	5.8 (10)	4.2 (10)	56.5 (10)	268.2 (10)
	200	0.2 (10)	0.3 (10)	0.5 (10)	6.6 (10)	38.3 (10)	0.2 (10)	0.2 (10)	0.6 (10)	5.3 (10)	92.4 (10)
	500	0.0 (10)	0.0 (10)	0.1 (10)	0.2 (10)	0.6 (10)	0.0 (10)	0.0 (10)	0.1 (10)	0.2 (10)	0.6 (10)
2000	1	1.7 (10)	4.2 (10)	12.8 (10)	111.6 (10)	1114.0 (9)	8.0 (10)	30.8 (10)	22.6 (10)	113.7 (10)	780.7 (9)
	200	1.1 (10)	4.3 (10)	11.2 (10)	95.7 (10)	249.7 (10)	2.2 (10)	10.4 (10)	10.5 (10)	174.8 (10)	1070.9 (9)
	500	0.1 (10)	0.1 (10)	0.2 (10)	1.1 (10)	4.2 (10)	0.1 (10)	0.2 (10)	0.2 (10)	1.3 (10)	5.1 (10)
3000	1	3.6 (10)	3.6 (10)	25.5 (10)	72.8 (10)	1792.8 (7)	27.7 (10)	29.1 (10)	210.9 (10)	707.5 (10)	2858.9 (4)
	200	2.5 (10)	12.6 (10)	71.2 (10)	719.2 (9)	1651.2 (7)	16.5 (10)	23.8 (10)	72.8 (10)	585.2 (9)	1853.9 (8)
	500	0.7 (10)	1.6 (10)	2.5 (10)	27.1 (10)	60.8 (10)	0.7 (10)	1.8 (10)	2.2 (10)	26.5 (10)	96.3 (10)
4000	1	65.4 (10)	13.4 (10)	97.0 (10)	963.8 (8)	1037.9 (8)	64.9 (10)	123.2 (10)	331.3 (10)	2000.2 (7)	3600.0 (0)
	200	32.0 (10)	22.5 (10)	21.0 (10)	1093.3 (9)	3079.3 (2)	51.7 (10)	86.5 (10)	185.7 (10)	1658.0 (8)	3103.8 (3)
	500	12.7 (10)	21.9 (10)	37.8 (10)	574.4 (10)	1698.1 (7)	33.6 (10)	59.3 (10)	138.3 (10)	408.4 (10)	1716.4 (8)
5000	1	207.5 (10)	248.4 (10)	356.8 (10)	1426.8 (7)	985.8 (9)	78.4 (10)	499.1 (10)	480.0 (10)	2686.0 (4)	3375.9 (2)
	200	12.6 (10)	162.7 (10)	77.3 (10)	1006.2 (9)	3149.7 (5)	140.4 (10)	500.4 (10)	563.4 (9)	1962.7 (7)	2905.2 (5)
	500	36.1 (10)	24.1 (10)	80.2 (10)	621.3 (10)	2540.7 (5)	112.1 (10)	46.9 (10)	120.4 (10)	1981.3 (7)	2857.0 (4)