Robust storage loading problems with stacking and payload constraints

Marc Goerigk[*1], Sigrid Knust[†2], and Xuan Thanh Le[‡2]

[1]Department of Mathematics, University of Kaiserslautern, Germany
[2]Department of Mathematics and Computer Science, University of Osnabrück, Germany

Abstract

We consider storage loading problems where items with uncertain weights have to be loaded into a storage area, taking into account stacking and payload constraints. Following the robust optimization paradigm, we propose strict and adjustable optimization models for finite and interval-based uncertainties. To solve these problems, exact decomposition and heuristic solution algorithms are developed. For strict robustness, we also present a compact formulation based on a characterization of worst-case scenarios. Computational results show that computation times and algorithm gaps are reasonable for practical applications. Furthermore, we find that the robustness concepts show different potential depending on the type of data being used.

Keywords: OR in maritime industry; storage loading; stacking problems; payload restrictions; robust optimization

1. Introduction

Storage loading problems appear in several practical applications, e.g., in the context of container terminals, container ships, warehouses or steel yards, see [26]. Especially container transportation plays a vital role in today's global economy. For convenient transportation by different modalities, containers are standardized in term of twenty-feet-equivalent-units (TEUs), which refers to containers of twenty-feet lengths. According to an executive summary in [32], the throughput of container ports all over the world was 651.1 million TEUs in 2013 (corresponding to 1524 millions tons of cargo), in which 160 million TEUs were transported by ships. Therefore, it can be seen that the transportation of seaborne containers contributes an important part in international trade. Furthermore, there has been a progressive increase in maximum container-ship size, from 1000 TEU ships by the mid-1970's to 18000 TEU vessels by 2015 (see [33]). Such numbers make professional and efficient operations essential in container ships. This motivates an active research topic on finding efficient solution methods for maritime container transportation. An up-to-date collection of planning and scheduling problems in large maritime container yards, together with latest solution approaches, can be found in [27]. Many other topics on maritime

[*]Email: goerigk@mathematik.uni-kl.de
[†]Email: sigrid.knust@uni-osnabrueck.de
[‡]Email: xuanthanh.le@uni-osnabrueck.de

container transportation, such as tactical and operational management in shipping liners, empty container repositioning, etc., are collected in [25].

In this paper, we focus on storage loading problems where incoming items arrive at a storage area and have to be assigned to stacks so that certain constraints are respected. Usually, only the topmost item of each stack can be directly retrieved, i.e., the items are accessed in last-in-first-out order. The items are often relocated by cranes moving above the stacks, which imposes a restriction on the maximum height of a stack. Additionally, certain stacking constraints have to be respected, i.e., not every item may be stacked on every other item. For example, heavier items are not allowed to be stacked on top of lighter ones, larger items are not allowed to be put on smaller ones, or items with a later departure time may not be stacked on items with an earlier departure time. For a survey and a classification scheme of such problems we refer to [26]. Complexity results for stacking problems taking into account a limited height of the stacks and stacking constraints are provided in [15].

In many practical loading problems, additional stability issues are crucial. In load planning of trains (cf. [14, 13]) containers have to be loaded onto wagons so that the stability of each wagon is guaranteed. In air cargo load planning (cf. [34]), which can be considered as an application of the one-dimensional balanced loading problem (cf. [4, 29]), a set of cargo has to be loaded on an aircraft such that the deviation between the aircraft's center of gravity and a target point is minimized (to improve stability of the aircraft and reduce fuel consumption). In three-dimensional container loading models (cf. [34] and related references therein), a set of rectangular boxes needs to be loaded into a container so that the total volume of the boxes loaded is maximized, with a restriction on the maximum number of boxes that can be stacked onto each other (due to the load bearing strength over the top face of each box).

Also, when loading items onto a ship, the stability of the ship is an important issue that needs to be taken into account. It follows from a physical principle on the relation between the gravity center and the equilibrium of an object (see [20], Chapter 12) that the position of the gravity center of the whole set of items on the ship affects the ship's stability: the lower the gravity center, the more stable the ship is. Ideally, to obtain the best stability of a ship, the items are stored in such a way that heavier items are assigned to lower levels. In the existing literature about storage loading problems in containerships, stability issues of the ships are usually handled by imposing such stacking constraints on the weights of the containers (i.e., heavier items must be put below lighter items). For example, such stacking constraints appear in the context of stowage planning, which is also known as the master bay plan problem (MBPP) (cf. [31, 2, 3]). Formally, this problem is to determine a plan of minimum operating time for stowing a set of containers of different types into available locations of a containership, with respect to some structural and operational constraints (e.g., a restriction on the maximum weight of the containership, containers retrieved later may not be stored on top of containers that are retrieved earlier). For the equilibrium of ships, the weights of containers are classified into three groups (light, medium, heavy), and the following restrictions are considered. First, the total weight of three consecutive containers in a stack cannot be greater than an a priori established value. Second, the weight on the right side of the ship should not differ much from the weight on the left side (for cross equilibrium). Finally, heavier containers are not allowed to be put on top of lighter ones (for horizontal equilibrium). As solution approaches, Sciomachen and Tanfani [31] present a heuristic method based on a relationship of the MBPP to three-dimensional bin packing problems. Ambrosino et al. [2] propose a binary linear programming model for the problem, present a heuristic approach, and give some prestowage rules for being able to solve the model. Ambrosino et al. [3] propose another solution approach for the same problem based on a three-phase algorithm using the idea of splitting the ship into different parts and assigning the containers to them on the basis of the containers' destinations.

In real-world containership loading problems, hard stacking constraints on the weights of the containers might be too conservative due to their interaction with other practical constraints. Therefore, in this paper we use another approach to control the stability of a ship and impose additional constraints on the payload of the items. More precisely, we assume that the total weight that can be put on top of an item i with weight w_i must be limited by aw_i, where a is a given positive parameter (this can also be used to handle fragile items). With this additional constraint on the payload of the items, the smaller the value of parameter a is, the lower the highest possible position of the gravity center of each stack is, and consequently, the more stable the ship is. The payload parameter a can therefore be used to control the stability of a ship – optimizing the center of gravity directly is a nonlinear objective and not the focus of this paper. An illustrative explanation of the relationship between the payload parameter a and the center of gravity is given in Appendix A.

In practice, it might also be possible that payload violations are allowed and the gravity center of the ship may be shifted to a higher position. To achieve the desired stability of the ship, an amount of ballast corresponding to the total payload violation is put at the bilge of the ship so that the gravity center of the whole ship is adjusted to a safe position (cf. [36]). By minimizing the total payload violation over all stacks, the amount of ballast needed is minimized, and consequently, the shipping cost is reduced while the ship's stability is guaranteed.

Since in real-world applications, often not all data are exactly known during the planning stage, in this paper, we consider storage loading problems under data uncertainty. In particular, we assume that the weight of each item is uncertain and may come either from a finite set of possible scenarios or from an interval of potential outcomes. We consider two approaches to include robustness in this setting: strict robustness (see [8]), where the location of each item needs to be fixed before its actual weight becomes known, and adjustable robustness (see [7]), where each item must only be assigned to a stack, but its position within the stack can be decided once the weight is known.

For general surveys on robust optimization, we refer to [19, 6, 23, 1, 10]. Other applications of robust optimization include load planning of trains [13], empty container repositioning [17], shunting problems [16], disaster management [18, 5], and many more. Storage loading problems with uncertain data have, for example, been studied by Kim et al. [22]. There, the weights of the items are classified into three groups and the weight group of each item is not known before its arrival. In the storage area, the heavier items should be stored in higher levels of the stacks, since they have to be retrieved earlier to put them in the bottom of a vessel (for stability reasons). Kang et al. [21] propose a simulated annealing algorithm to find a good stacking strategy for a similar problem where the incoming items have uncertain weights. In [24], incoming items arriving at a partly filled storage area have to be assigned to stacks regarding that not every item may be stacked on top of every other item and taking into account uncertain data of items arriving later. Strict and adjustable robust solutions are calculated using different MIP formulations.

The remainder of the paper is organized as follows. In Section 2, we describe the deterministic stacking problem and formally introduce the uncertainty sets. The strictly robust counterpart of this uncertain problem is considered for both finite and interval uncertainty sets in Section 3, while adjustable counterparts are discussed in Section 4. Computational experiments are presented in Section 5. Finally, conclusions can be found in Section 6.

2. Problem formulation

In this section, we give a formal definition of the studied storage problem, formulate its deterministic version as a mixed-integer program (MIP), and introduce the considered uncertainties.

2.1. Nominal problem

In the following, we describe the nominal (deterministic) problem in more detail. Let m be the number of stacks where for each stack a position in the storage area is fixed. Each stack can hold at most b items. The set of all items is denoted by $I = \{1, 2, \ldots, n\}$ where normally the inequality $m < n$ holds, i.e. some items have to be stacked on others. Since all items have to be stacked, we assume that $n \leq bm$; otherwise the problem is infeasible.

As a hard constraint we assume that not every item may be stacked on every other item (for example, a larger item may not be stacked on top of a smaller one or an item with a later departure time may not be stacked on top of one with an earlier departure time). Such stacking constraints may be encoded by a 2-dimensional binary matrix $S = (s_{ij})_{n\times n}$, where $s_{ij} = 1$ if i can be stacked onto j and $s_{ij} = 0$ otherwise.

Items stored in a stack are defined by a tuple $(i_k, \ldots, i_1)$, where i_l denotes the item stacked at level l and $l = 1$ corresponds to the ground level. Such a tuple is feasible if $k \leq b$ and $s_{i_{l+1},i_l} = 1$ for all $l = 1, \ldots, k-1$.

Additionally, we assume that each item $i \in I$ has a weight w_i and that the total weight of items put on top of item i should not be larger than aw_i with a given payload factor $a \in \mathbb{R}_{\geq 0}$. If the total weight W of all items above i exceeds aw_i, a payload violation of $W - aw_i$ occurs. The total payload violation of a stacking configuration is defined as the sum of the payload violations over all items in all stacks of the configuration. In this paper, the payload constraints are assumed to be soft, i.e., payload violations are allowed, but have to be minimized. Our discussion also includes the situation of hard payload constraints, since in this case a feasible solution exists if and only if the minimal total payload violation is equal to zero.

The simplest version of a storage loading problem is the feasibility problem (cf. [15]) which asks whether all items can be feasibly allocated to the storage area respecting all hard constraints, i.e. the stack capacity b and the stacking constraints s_{ij}. If this is possible, the objective is to assign each item to a feasible location (specified by a stack number and a level in the stack). In [15] it was shown that deciding whether a feasible solution exists is strongly NP-complete even for $b = 3$ and transitive stacking constraints. In an optimization version of the problem additionally some objective function (e.g., the total number of used stacks, the number of items stacked above the ground level or the number of misordered items with respect to departure times) may be minimized. In this paper we concentrate on minimizing the total payload violation as objective function. This problem is strongly NP-hard for $b \geq 3$, since it generalizes the feasibility problem.

2.2. A MIP formulation

In the following, we present a MIP formulation for the nominal problem where $w \in \mathbb{R}^n$ is the vector of the nominal weights of all items. Furthermore, let $Q := \{1, \ldots, m\}$ be the set of stacks and $L := \{1, \ldots, b\}$ be the set of levels. We use the notation $[\alpha]_+$ to indicate $\max\{\alpha, 0\}$. Let x_{iql} for $i \in I, q \in Q, l \in L$ be binary variables with

$$x_{iql} = \begin{cases} 1, & \text{if item } i \text{ is stored in stack } q \text{ at level } l, \\ 0, & \text{otherwise.} \end{cases}$$

For a stacking configuration encoded by x, the payload violation of an item in stack $q \in Q$ at level $l \in L \setminus \{b\}$ is

$$\left[\sum_{j\in I} \sum_{h=l+1}^{b} w_j x_{jqh} - a \sum_{i \in I} w_i x_{iql} \right]_+ ,$$

and hence the total payload violation of the configuration is given by

$$f(x,w) := \sum_{q\in Q}\sum_{l\in L\setminus\{b\}} \left[\sum_{j\in I}\sum_{h=l+1}^{b} w_j x_{jqh} - a\sum_{i\in I} w_i x_{iql}\right]_+ .$$

To linearly represent the objective function $f(x,w)$, we use additional non-negative variables v_{ql} for $q \in Q, l \in L$ to compute the payload violation of the item stored in stack q at level l. Then the problem can be formulated as follows.

$$\min \sum_{q\in Q}\sum_{l\in L\setminus\{b\}} v_{ql} \tag{1}$$

$$\text{s.t.} \sum_{q\in Q}\sum_{l\in L} x_{iql} = 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{i\in I} x_{iql} \le 1 \qquad \forall q \in Q, l \in L \tag{3}$$

$$\sum_{j\in I\setminus\{i\}} s_{ij} x_{jq,l-1} \ge x_{iql} \qquad \forall i \in I, q \in Q, l \in L \setminus \{1\} \tag{4}$$

$$\sum_{j\in I}\sum_{h=l+1}^{b} w_j x_{jqh} - a\sum_{i\in I} w_i x_{iql} \le v_{ql} \qquad \forall q \in Q, l \in L \setminus \{b\} \tag{5}$$

$$x_{iql} \in \{0,1\} \qquad \forall i \in I, q \in Q, l \in L \tag{6}$$

$$v_{ql} \ge 0 \qquad \forall q \in Q, l \in L \setminus \{b\} \tag{7}$$

According to (1) the sum of all payload violations is minimized. Constraints (2) guarantee that all items are stored. Constraints (3) ensure that at most one item is stored at each level of each stack. Due to (4) the stacking constraints s_{ij} are satisfied and no item is placed to a location where no item is stacked below. Inequalities (5) ensure that the payload violations v_{ql} are computed correctly.

We refer to this problem as (P). Note that (P) uses $\mathcal{O}(nmb)$ many variables and constraints. In the following we denote by

$$\mathcal{X} := \left\{x \in \{0,1\}^{|I|\times|Q|\times|L|} \mid x \text{ satisfies (2)-(4)}\right\}$$

the set of feasible solutions respecting all hard constraints of the stacking problem.

2.3. Uncertainties

In this paper, we consider two kinds of uncertainties for the item weights that affect the payload constraints:

- In the first case, we assume that for every item i, we are provided with lower and upper bounds $\underline{w}_i$, $\overline{w}_i$ on the possible outcome of item weights. We write

  $$\mathcal{U}^I = [\underline{w}_1, \overline{w}_1] \times \ldots \times [\underline{w}_n, \overline{w}_n]$$

  to denote an interval-based uncertainty set. An element $w \in \mathcal{U}^I$ is called a scenario. The lower and upper bounds stem from empirical observations or expert knowledge. We do not assume knowledge of any probability distribution over $\mathcal{U}^I$.

- In the second case, we assume that we are given a list of N possible scenarios, where as before a scenario consists of a weight for each item. We write

$$\mathcal{U}^F = \{w^1, \dots, w^N\}$$

for the uncertainty set containing all possible outcomes. Such a description of scenarios may either be based on the expertise of practitioners (e.g., an experienced storage loading manager is able to enumerate typical outcomes of uncertain weights), or may stem from a probabilistic analysis and represents the most likely outcomes. We write $\mathcal{N} = \{1, \dots, N\}$.

In case that we do not need to distinguish these two kinds of uncertainty sets, we simply write $\mathcal{U}$ to denote the uncertainty set. Note that $\mathcal{U}^F$ is a finite set, while $\mathcal{U}^I$ contains infinitely many possible outcomes. This leads to different solution approaches for the robust models we consider in this paper.

3. Strict robustness

In this section, we consider the problem setting where a complete stacking solution has to be fixed in advance before the actually realized scenario becomes known. Such an approach is required if the storage plan has to be announced before the actual weights of the items are known and the plan cannot be changed later on. This means that the planner has to find a complete stacking solution, i.e., to decide for each item to which stack and level it is assigned, based on incomplete knowledge. Following the approach of [23, 8], we focus on strictly robust solutions where the worst-case payload violation over all scenarios is minimized. The strictly robust counterpart (SR, $\mathcal{U}$) of the optimization storage loading problem (P) under affection of uncertainty set $\mathcal{U}$ is

$$(\text{SR}, \mathcal{U}) \qquad \min_{x \in \mathcal{X}} \max_{w \in \mathcal{U}} f(x, w).$$

We first consider the case of finite uncertainty in Section 3.1, afterwards the more elaborate case of interval-based uncertainty is discussed in Section 3.2.

3.1. Finite uncertainty

In the following, we modify the problem formulation (P) to include a finite uncertainty set. First, we introduce new variables $v_{ql}^k \geq 0$ for $q \in Q, l \in L \setminus \{b\}, k \in \mathcal{N}$ measuring the payload violation of the item stored in stack q at level l in the solution for scenario k. Additionally, an auxiliary variable $v \geq 0$ is introduced to measure the total payload violation in the worst-case over all scenarios. We denote this problem as (SRF), though we may also write (SR, $\mathcal{U}^F$) when the usage of uncertainty set $\mathcal{U}^F$ should be emphasized, and obtain the following MIP formulation:

$$(\text{SRF}) \qquad \min v \tag{8}$$

$$\sum_{q \in Q} \sum_{l \in L} x_{iql} = 1 \qquad \forall i \in I \tag{9}$$

$$\sum_{i \in I} x_{iql} \leq 1 \qquad \forall q \in Q, l \in L \tag{10}$$

$$\sum_{j \in I \setminus \{i\}} s_{ij} x_{jq,l-1} \geq x_{iql} \qquad \forall i \in I, q \in Q, l \in L \setminus \{1\} \tag{11}$$

$$\sum_{j \in I} \sum_{h=l+1}^{b} w_j^k x_{jqh} - a \sum_{i \in I} w_i^k x_{iql} \leq v_{ql}^k \qquad \forall q \in Q, l \in L \setminus \{b\}, k \in \mathcal{N} \tag{12}$$

$$\sum_{q\in Q}\sum_{l\in L\setminus\{b\}} v_{ql}^k \le v \qquad \forall k \in \mathcal{N} \tag{13}$$

$$x_{iql} \in \{0,1\} \qquad \forall i \in I, q \in Q, l \in L \tag{14}$$

$$v_{ql}^k \ge 0 \qquad \forall q \in Q, l \in L \setminus \{b\}, k \in \mathcal{N} \tag{15}$$

$$v \ge 0 \tag{16}$$

The objective (8) is to minimize the largest total payload violation over all scenarios. As in the nominal problem, constraints (9) ensure that every item is stored, constraints (10) model that at most one item is assigned to every location, and constraints (11) take care of the stacking constraints s_{ij}. Furthermore, according to constraints (12) the payload violations v_{ql}^k are correctly computed for every scenario k. Finally, constraints (13) (together with the minimization in (8)) ensure that the variable v equals the maximum of these values. Compared to the nominal model (which uses $\mathcal{O}(nmb)$ many variables and constraints), this formulation requires $\mathcal{O}((n+N)mb)$ many variables and constraints.

3.2. Interval uncertainty

We now consider (SR, $\mathcal{U}^I$), i.e., the strictly robust model with interval-based uncertainty sets $\mathcal{U}^I$. We denote this problem as (SRI) for short. Due to the continuous nature of the uncertainty set $\mathcal{U}^I$, there are infinitely many scenarios and we cannot include all these scenarios into a MIP formulation as we have done with (SRF). Therefore, we discuss approaches that iteratively choose scenarios from $\mathcal{U}^I$ to include them in a finite uncertainty set (see also [30, 12, 35], where similar ideas have successfully been applied). The general procedure is shown in Algorithm 1. We start with an arbitrary scenario $w^0 \in \mathcal{U}^I$. In each iteration k, we find a best stacking configuration x^k with respect to the current (finite) set of scenarios $\mathcal{U}^k$, i.e., x^k is a solution to (SR, $\mathcal{U}^k$). Then we determine a worst-case scenario $w^{k+1} \in \mathcal{U}^I$ corresponding to this stacking configuration (i.e., a scenario of item weights maximizing the total payload violation $f(x^k, w)$ for configuration x^k). The worst-case scenario found in each step is added to the current set of scenarios. This is repeated until the objective value of the robust problem and the objective value of the worst-case problem coincide.

Algorithm 1 Exact algorithm for (SRI)

```
Require: An instance of (SRI).
 1: $k \leftarrow 0$
 2: Take an arbitrary scenario $w^0 \in \mathcal{U}^I$ and let $\mathcal{U}^0 \leftarrow \{w^0\}$.
 3: Solve (SR, $\mathcal{U}^k$). Let $x^k$ be the resulting stacking solution and $LB^k$ its objective value.
 4: Find a scenario $w^{k+1} \in \mathcal{U}^I$ that maximizes the total payload violation for stacking configu-
    ration $x^k$. Let $UB^k$ be the corresponding (worst-case) total payload violation.
 5: if $UB^k = LB^k$ then
 6:     return optimal stacking solution $x^k$
 7: else
 8:     $\mathcal{U}^{k+1} \leftarrow \mathcal{U}^k \cup \{w^{k+1}\}$
 9:     $k \leftarrow k+1$
10:     Goto 3
11: end if
```

Theorem 1 *Algorithm 1 terminates after a finite number of iterations and yields an optimal solution x to (SRI).*

Proof: Let f^* be the optimal objective value of (SRI), i.e.,

$$f^* = \min_{x \in \mathcal{X}} \max_{w \in \mathcal{U}^I} f(x, w).$$

Since $\mathcal{U}^k \subseteq \mathcal{U}^{k+1} \subset \mathcal{U}^I$, we have

$$\begin{aligned} LB^k &= \min_{x \in \mathcal{X}} \max_{w \in \mathcal{U}^k} f(x, w) \\ &\leq \min_{x \in \mathcal{X}} \max_{w \in \mathcal{U}^{k+1}} f(x, w) = LB^{k+1} \\ &\leq \min_{x \in \mathcal{X}} \max_{w \in \mathcal{U}^I} f(x, w) = f^*. \end{aligned}$$

On the other hand, by definition of UB^k in Step 4 of the algorithm, we have

$$UB^k = f(x^k, w^{k+1}) = \max_{w \in \mathcal{U}^I} f(x^k, w) \geq \min_{x \in \mathcal{X}} \max_{w \in \mathcal{U}^I} f(x, w) = f^*.$$

This means that LB^k is a lower bound on f^* and UB^k is an upper bound on f^*. Therefore, if $LB^k = UB^k$, then (x^k, w^{k+1}) is an optimal solution to (SRI) and $LB^k = f^*$ is the optimal objective value of (SRI).

We now show that if $LB^k \neq UB^k$, then $w^{k+1} \notin \mathcal{U}^k$, so that the algorithm never enters a cyclic loop. Indeed, assume to the contrary that $w^{k+1} \in \mathcal{U}^k$. As defined in Step 4 of the algorithm, we have

$$\begin{aligned} & w^{k+1} = \operatorname*{argmax}_{w \in \mathcal{U}^I} f(x^k, w) \\ \Leftrightarrow \quad & UB^k = f(x^k, w^{k+1}) \geq f(x^k, w) \quad \forall w \in \mathcal{U}^I \\ \Rightarrow \quad & UB^k = f(x^k, w^{k+1}) \geq f(x^k, w) \quad \forall w \in \mathcal{U}^k && (\text{since } \mathcal{U}^k \subset \mathcal{U}^I) \\ \Leftrightarrow \quad & UB^k = f(x^k, w^{k+1}) = \max_{w \in \mathcal{U}^k} f(x^k, w) && (\text{since } w^{k+1} \in \mathcal{U}^k). \end{aligned}$$

Moreover, as defined in Step 3 of the algorithm, we have $LB^k = \max_{w \in \mathcal{U}^k} f(x^k, w)$. Therefore, we again obtain $LB^k = UB^k$ under the assumption that $w^{k+1} \in \mathcal{U}^k$. This means that if $LB^k \neq UB^k$, then we must have $w^{k+1} \notin \mathcal{U}^k$.

The termination of the algorithm after a finite number of iterations follows immediately from the two following claims: (a) the number of possible stacking configurations x^k generated by the algorithm is finite, (b) the number of possible worst-case scenarios w^k generated by the algorithm is also finite. Indeed, since there is a finite number of items, also the number of possible stacking configurations for these items is finite, and claim (a) follows. By Step 4 of the algorithm, we generate only one worst-case scenario w^{k+1} corresponding to stacking configuration x^k, so claim (b) follows from claim (a). $\square$

How to solve Step 3 of the algorithm was shown in Section 3.1. We now discuss how Step 4 can be realized. Given a stacking configuration x, we need to find a vector $w \in \mathcal{U}^I$ of item weights maximizing the total payload violation (i.e., the sum of payload violations over all levels of all stacks). Since there is no payload violation in stacks containing only one item, we have to compute the maximum total payload violation in all stacks containing at least two items.

For any stack $q \in Q$ in the given configuration x, let $I(q)$ be the set of items contained in it and $L(q) := \{1, \ldots, |I(q)|\}$. We assume that $|I(q)| \geq 2$. For the sake of simplicity, we denote the weight of the item at level $l \in L(q)$ by $w_{[l]}$.

Lemma 2 *The total payload violation of any stack $q \in Q$, given by*

$$v_q(w) = \sum_{l \in L(q)} \left[\sum_{h>l} w_{[h]} - a w_{[l]} \right]_+$$

is a convex function.

Proof: We have that $\sum_{h>l} w_{[h]} - a w_{[l]}$ is linear in w, and taking the maximum of two convex functions is again a convex function. Since also the sum of convex functions is convex, the claim follows. □

We denote the problem of finding item weights w that maximize the payload violation $v_q(w)$ for a stack $q \in Q$ as (V). Due to the convexity of v_q, we can make use of the fact that there is always an optimal solution to (V) where each item weight is at its lower or upper bound. This gives rise to the following formulation as a mixed-integer program. For all $l \in L(q)$, we introduce continuous variables $w_{[l]}$ determining the weight of the item at level l in q. Furthermore, we use binary variables β_l with $\beta_l = 0$ if $w_{[l]} = \underline{w}_{[l]}$ and $\beta_l = 1$ if $w_{[l]} = \overline{w}_{[l]}$. Finally, variables α_l are used to correctly compute the payload violation $[\sum_{h>l} w_{[h]} - a w_{[l]}]_+$ in the objective function. We have $\alpha_l = 1$ if $\sum_{h>l} w_{[h]} - a w_{[l]} \geq 0$ and $\alpha_l = 0$ if $\sum_{h>l} w_{[h]} - a w_{[l]} < 0$. Then, problem (V) can be formulated as follows.

$$\text{(V)} \quad \max \sum_{l \in L(q)} \left(\sum_{h>l} w_{[h]} - a w_{[l]} \right) \alpha_l \tag{17}$$

$$\text{s.t.} \quad w_{[l]} = \underline{w}_{[l]} + (\overline{w}_{[l]} - \underline{w}_{[l]}) \beta_l \qquad \forall l \in L(q) \tag{18}$$

$$\alpha_l, \beta_l \in \{0,1\} \qquad \forall l \in L(q) \tag{19}$$

$$w_{[l]} \geq 0 \qquad \forall l \in L(q) \tag{20}$$

Note that this formulation is non-linear, due to the product of α and w in the objective function. We can remove variables $w_{[l]}$ by inserting equations (18) in the objective function, and get the equivalent model

$$\begin{aligned} \max & \sum_{l \in L(q)} \sum_{h>l} \left(\underline{w}_{[h]} + \left(\overline{w}_{[h]} - \underline{w}_{[h]} \right) \beta_h \right) \alpha_l \\ & - \sum_{l \in L(q)} a \left(\underline{w}_{[l]} + \left(\overline{w}_{[l]} - \underline{w}_{[l]} \right) \beta_l \right) \alpha_l \end{aligned} \tag{21}$$

$$\text{s.t.} \quad \alpha_l, \beta_l \in \{0,1\} \qquad \forall l \in L(q) \tag{22}$$

By introducing new variables $\gamma_{lh} = \alpha_l \cdot \beta_h$ for all $l, h \in L(q)$, we obtain the binary linear program

$$\begin{aligned} \max & \sum_{l \in L(q)} \left(\sum_{h>l} \underline{w}_{[h]} - a \underline{w}_{[l]} \right) \alpha_l \\ & + \sum_{l \in L(q)} \left(\sum_{h>l} (\overline{w}_{[h]} - \underline{w}_{[h]}) \gamma_{lh} - a (\overline{w}_{[l]} - \underline{w}_{[l]}) \gamma_{ll} \right) \end{aligned} \tag{23}$$

$$\text{s.t.} \quad \alpha_l + \beta_h - 1 \leq \gamma_{lh} \leq \frac{1}{2}(\alpha_l + \beta_h) \qquad \forall l, h \in L(q) \tag{24}$$

$$\alpha_l, \beta_l \in \{0,1\} \qquad \forall l \in L(q) \tag{25}$$

$$\gamma_{lh} \in \{0,1\} \qquad \forall l, h \in L(q) \tag{26}$$

The objective function (23) is the same as in (21) after substituting and reordering terms. The additional constraints (24) are used to ensure that γ_{lh} is one if and only if both α_l and β_h are one. Solving problem (23)-(26) independently for each stack $q \in Q$ hence gives the desired solution to Step 4 of Algorithm 1.

Note that maximizing a convex function over a convex domain in general is an NP-hard problem [9]. However, we can show that for this special case an efficient solution algorithm exists.

Theorem 3 *For a given stacking solution and any value of the payload parameter a, the maximum total payload violation of each stack $q \in Q$ can be found by evaluating $\mathcal{O}(|I(q)|^{2\delta-1})$ scenarios, where $\delta := \min\{\lceil a \rceil, \lfloor \frac{|I(q)|}{2} \rfloor\}$.*

Proof: As mentioned above, due to the convexity of v_q, for finding an optimal solution to (V) it is sufficient to consider only scenarios where the weights of all items in $I(q)$ are either on their lower or upper bounds. For a choice of item weights w, we say that at level $l < |I(q)|$ a solution has a *break* if $w_{[l]} = \underline{w}_{[l]}$ and $w_{[l+1]} = \overline{w}_{[l+1]}$, and an *anti-break* if $w_{[l]} = \overline{w}_{[l]}$ and $w_{[l+1]} = \underline{w}_{[l+1]}$. Note that there is alway an optimal solution where the bottom item is as light as possible, and the top item is as heavy as possible; therefore, there is always an optimal solution with at least one break. Whenever there is a break at some level l of an optimal solution to (V), without loss of generality we can assume that $\sum_{h>l} w_{[h]} - aw_{[l]} \geq 0$. Indeed, if this was not the case, then we have $\sum_{h>l} w_{[h]} < aw_{[l]}$, i.e., there is no payload violation at level l of stack q. Therefore there is still no payload violation at this location if we increase $w_{[l]}$ by a positive amount. In other words, we could increase the weight of the item at level l without decreasing $v_q(w)$.

We now show that there is an optimal solution to (V) with at most δ breaks. Firstly, we note that each break occupies two consecutive levels in the stack, and different breaks occupy different levels. Therefore, there are no more than $\lfloor \frac{|I(q)|}{2} \rfloor$ breaks in stack q. Secondly, there exists an optimal solution to (V) with at most $\lceil a \rceil$ breaks. Indeed, let w^* be some maximizer of (V) with $\beta^* > \lceil a \rceil$ breaks and let l^* be the level of the topmost break. If we increase the weight of the item at level l^* by $\Delta = \overline{w}_{[l^*]} - \underline{w}_{[l^*]}$, the payload violation at level l^* decreases by at most $a\Delta$. However, there are at least $\beta^* - 1$ more payload violations beneath level l^*, which result in an increase of $v_q(w)$ by at least $(\beta^* - 1)\Delta$. Due to $\beta^* > \lceil a \rceil$, the total payload violation could be increased. Therefore, we can remove the break at level l^* without decreasing the violation $v_q(w)$. Repeating this argument until the next break level, we find that there is an optimal solution with at most $\lceil a \rceil$ breaks.

We now count the number of possible scenarios with $\beta \in \{1, \ldots, \delta\}$ breaks. In such a scenario, between any two consecutive breaks there must be exactly one anti-break. Therefore, each of such scenarios corresponds to a choice of $2\beta - 1$ levels for β breaks together with $\beta - 1$ anti-breaks in between. Since stack q contains $|I(q)|$ items, there are $|I(q)| - 1$ levels that can have breaks or anti-breaks. This leads to $\binom{|I(q)|-1}{2\beta-1}$ possible scenarios having β breaks. Enumerating all these possibilities for $\beta = 1, 2, \ldots, \delta$ gives $\mathcal{O}(|I(q)|^{2\delta-1})$ possible scenarios that have to be tested. □

Note that if a is a fixed value, the complexity $\mathcal{O}(|I(q)|^{2\delta-1})$ is polynomially bounded in the input length of the problem and hence (V) can be solved in polynomial time.

Due to Theorem 3, Step 4 of Algorithm 1 can alternatively be realized by enumerating all relevant item weights per level. However, the result can also be used to avoid the iterative algorithm and to formulate a compact model that includes all relevant scenarios directly. Note that these are not scenarios in the sense that a specific item gets some weight; instead, we assign to a specific level either the lower or upper weight of an item. The result is a formulation similar to the one used in Section 3.1.

We present this compact formulation of (SRI) for the case $a \leq 1$ where we know that for each stack only a single break has to be considered. Hence, only $|I(q)| - 1$ scenarios are relevant for stack q, where in the solution for scenario k the break occurs at level k. We introduce auxiliary variables $w_{ql}^k \geq 0$ denoting the weight of the item in stack q at level l in the solution for scenario k with a break at level k. Modifying the formulation (SRF), we get the following MIP formulation for (SRI):

$$
\begin{aligned}
& \min \ v && && (27)\\
\text{s.t.} \quad & \sum_{q\in Q}\sum_{l\in L} x_{iql} = 1 && \forall i \in I && (28)\\
& \sum_{i\in I} x_{iql} \leq 1 && \forall q \in Q, l \in L && (29)\\
& \sum_{j\in I\setminus\{i\}} s_{ij} x_{jq,l-1} \geq x_{iql} && \forall i \in I, q \in Q, l \in L \setminus \{1\} && (30)\\
& w_{ql}^k = \sum_{i\in I} \underline{w}_i x_{iql} && \forall q \in Q, k, l \in L, l \leq k && (31)\\
& w_{ql}^k = \sum_{i\in I} \overline{w}_i x_{iql} && \forall q \in Q, k, l \in L, l > k && (32)\\
& \sum_{h>l} w_{qh}^k - a w_{ql}^k \leq v_{ql}^k && \forall q \in Q, k, l \in L \setminus \{b\} && (33)\\
& \sum_{q\in Q}\sum_{l\in L\setminus\{b\}} v_{ql}^k \leq v && \forall k \in L\setminus\{b\} && (34)\\
& x_{iql} \in \{0,1\} && \forall i \in I, q \in Q, l \in L && (35)\\
& v_{ql}^k \geq 0 && \forall q \in Q, k, l \in L \setminus \{b\} && (36)\\
& w_{ql}^k \geq 0 && \forall q \in Q, k, l \in L && (37)\\
& v \geq 0 && && (38)
\end{aligned}
$$

In the solution for scenario k, all items up to level k are assumed to be as light as possible (constraints (31)), while all items at higher levels are as heavy as possible (constraints (32)). The payload violation in the solution for scenario k for stack q and level l is measured by the variable v_{ql}^k in constraints (33). The worst-case over all scenarios is computed with the help of the variable v and constraints (34).

For $a > 1$ a similar formulation can be used by enumerating all possible scenarios via the number of breakpoints, using $\mathcal{O}(b^{2\delta-1})$ relevant scenarios per stack. We present the formulation for $a \leq 2$ in Appendix B.

Note that in the case of hard payload constraints (i.e., no payload violations are allowed which implies that the total payload violation must be equal to zero), a more compact formulation can be given. For this, we consider the single robust payload constraint

$$\sum_{j\in I}\sum_{h=l+1}^{b} w_j x_{jqh} - a \sum_{i\in I} w_i x_{iql} \leq 0 \qquad \forall q \in Q, l \in L \setminus \{b\}, w \in \mathcal{U}^I$$

For a fixed location $(q,l) \in Q \times (L \setminus \{b\})$, the first term of the left hand side in this inequality is equal to the total weight of all items stored above this location, while the sum in the second term equals the weight of the item stored at the location. The maximum difference over all $w \in \mathcal{U}^I$ between the former and the latter term is therefore attained when the item stored at the location has minimum weight, while the items stored above have maximum weights. As we have pointed out, there exists a worst-case scenario $w \in \mathcal{U}^I$ which dominates all other scenarios from $\mathcal{U}^I$ with respect to this constraint. Hence, this robust payload constraint can be equivalently written as

$$\sum_{j\in I}\sum_{h=l+1}^{b} \overline{w}_j x_{jqh} - a\sum_{i\in I} \underline{w}_i x_{iql} \leq 0 \qquad \forall q \in Q, l \in L\setminus\{b\} \tag{39}$$

Now, any stacking solution has zero payload violation if and only if all these worst-case inequalities are fulfilled, i.e., the robust counterpart of the problem is given by (1)-(4),(39),(6).

4. Adjustable robustness

Following the ideas first introduced in [7], we now consider a robust model where not all stacking decisions need to be fixed in advance, but some can be made after the realized scenario becomes known. In our setting, we follow the idea that a planner needs to determine in advance to which stack an item is assigned ("here-and-now" decision); however, he is allowed to choose the level of the item within the stack depending on the weight scenario of all items later ("wait-and-see" decision). This gives the planner more flexibility in his decision making and potentially better results with less payload violations. Such a setting occurs in practice if special subareas (stacks) must be reserved for items in advance.

In Section 4.1, we first consider finite uncertainty sets before the more complex interval-based models are discussed in Section 4.2.

4.1. Finite uncertainty

As in Section 3, we would like to optimize the worst-case performance of a stacking solution over all possible weight realizations. There are two kinds of decisions that need to be made: Here-and-now decisions independent of realized item weights, which determine for every item the stack it is assigned to; and wait-and-see decisions depending on the scenario, which decide the level of each item at which it should be stored.

We introduce binary here-and-now variables z_{iq} for $i \in I, q \in Q$ where $z_{iq} = 1$ if item i is assigned to stack q. Furthermore, wait-and-see variables x^k_{iql} depend on the realized scenario k and determine if item i is stored in stack q at level l in the solution corresponding to scenario k. Then the problem can be formulated as follows.

$$\text{(ARF)} \quad \min\ v \tag{40}$$

$$\text{s.t.} \sum_{q\in Q} z_{iq} = 1 \qquad \forall\, i \in I \tag{41}$$

$$\sum_{i\in I} z_{iq} \leq b \qquad \forall\, q \in Q \tag{42}$$

$$\sum_{q\in Q}\sum_{l\in L} x^k_{iql} = 1 \qquad \forall\, i \in I, k \in \mathcal{N} \tag{43}$$

$$\sum_{i\in I} x^k_{iql} \leq 1 \qquad \forall\, q \in Q, l \in L, k \in \mathcal{N} \tag{44}$$

$$\sum_{l\in L} x^k_{iql} = z_{iq} \qquad \forall\, q \in Q, i \in I, k \in \mathcal{N} \tag{45}$$

$$\sum_{j\in I\setminus\{i\}} s_{ij} x^k_{jq,l-1} \geq x^k_{iql} \qquad \forall\, i \in I, q \in Q, l \in L\setminus\{1\} \tag{46}$$

$$\sum_{j\in I}\sum_{h=l+1}^{b} w_j^k x_{jqh}^k - a\sum_{i\in I} w_i^k x_{iql}^k \leq v_{ql}^k \qquad \forall\, q\in Q, l\in L\setminus\{b\}, k\in\mathcal{N} \tag{47}$$

$$\sum_{q\in Q}\sum_{l\in L\setminus\{b\}} v_{ql}^k \leq v \qquad \forall\, k\in\mathcal{N} \tag{48}$$

$$z_{iq}\in\{0,1\} \qquad \forall\, i\in I, q\in Q \tag{49}$$

$$x_{iql}^k\in\{0,1\} \qquad \forall\, i\in I, q\in Q, l\in L, k\in\mathcal{N} \tag{50}$$

$$v_{ql}^k \geq 0 \qquad \forall\, q\in Q, l\in L\setminus\{b\}, k\in\mathcal{N} \tag{51}$$

$$v\geq 0 \tag{52}$$

Constraints (41) model that every item has to be assigned to some stack, while every stack contains at most b items (constraints (42)). Constraints (43) ensure that also for each scenario every item has to be assigned to exactly one stack and level in the corresponding solution, while constraints (44) restrict the number of items at any location to be at most one. We couple the here-and-now variables z_{iq} with the wait-and-see variables x_{iql}^k in constraints (45): If the here-and-now decisions assign item i to stack q, then also in every scenario k item i has to be assigned to some level l in stack q. Constraints (46)-(48) are used to model the stacking constraints and to compute the payload violations.

Note that this formulation is more sensitive to the number of scenarios than the strictly robust model, with $\mathcal{O}(Nnmb)$ many variables and $\mathcal{O}((N+n)mb)$ many constraints.

4.2. Interval uncertainty

We now consider the adjustable robust problem with interval uncertainty, denoted as (ARI, $\mathcal{U}^I$). As in Section 3.2, we follow the idea of an iterative approach that considers relaxed problems with finite uncertainty sets, and a subproblem to generate worst-case scenarios. This subproblem now becomes more complex, since the worst-case generation also needs to take the possible wait-and-see decisions into account.

According to the here-and-now decisions all items are assigned to stacks, but their ordering in the stacks (i.e., the assignment to levels) is determined later when the weights of all items are known. To this end, we consider the following two subproblems for a single stack:

1. Given a subset of possible item orderings, we search for a worst-case weight scenario that maximizes the smallest total payload violation over all orderings.
2. Given a weight scenario, we search for an item ordering that minimizes the total payload violation.

Both subproblems are iteratively solved, until their objective values coincide.

We consider a fixed stack $q\in Q$ and assume that the set $I(q)$ contains all items which are assigned to q according to the here-and-now decisions z_{iq}. Furthermore, let $L(q) := \{1,\ldots,|I(q)|\}$ and $L'(q) := L(q)\setminus\{|I(q)|\}$. Let $\mathcal{P}(I_q)$ be the set of all permutations for the items in the set $I(q)$, each permutation describing an assignment of all items to levels in the stack. Furthermore, we denote by $v_q(\pi,w)$ the total payload violation for stack q with respect to the weights w if the items in $I(q)$ are ordered according to the permutation π.

In the first subproblem, for a given subset $\mathcal{P}'(I_q)\subseteq\mathcal{P}(I_q)$ of permutations for the items in stack q we search for a worst-case weight scenario $w\in\mathcal{U}^I$ that is a solution of problem

$$(\text{Max-}w) \qquad \max_{w\in\mathcal{U}^I}\ \min_{\pi\in\mathcal{P}'(I_q)} v_q(\pi,w).$$

In the following, we assume that the set $\mathcal{P}'(I_q)$ contains K permutations $\pi^1, \dots, \pi^K$ which are described by binary values p_{il}^k with $p_{il}^k = 1$ if and only if in permutation π^k item $i \in I(q)$ is assigned to level $l \in L(q)$. Let $\mathcal{K} := \{1, \dots, K\}$.

To solve problem (Max-w) for all $i \in I(q)$, we introduce continuous variables $w_i \in [\underline{w}_i, \overline{w}_i]$ determining the weight of item i. Additionally, we have variables $v_l^k \geq 0$ measuring the payload violation for the item assigned to level l in the permutation π^k and binary auxiliary variables α_l^k to determine whether there is a payload violation at level l in the permutation π^k or not. Finally, the auxiliary variable $v \geq 0$ denotes the total payload violation of the stack. Then problem (Max-w) can be formulated as follows.

$$\max \ v \tag{53}$$

$$\text{s.t.} \sum_{l \in L'(q)} v_l^k \geq v \qquad \forall k \in \mathcal{K} \tag{54}$$

$$\left(\sum_{h>l} \sum_{i \in I(q)} p_{ih}^k w_i - a \sum_{i \in I(q)} p_{il}^k w_i \right) \alpha_l^k = v_l^k \qquad \forall k \in \mathcal{K}, l \in L'(q) \tag{55}$$

$$\underline{w}_i \leq w_i \leq \overline{w}_i \qquad \forall i \in I(q) \tag{56}$$

$$\alpha_l^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, l \in L'(q) \tag{57}$$

$$v_l^k \geq 0 \qquad \forall k \in \mathcal{K}, l \in L'(q) \tag{58}$$

$$v \geq 0 \tag{59}$$

According to (53) the total payload violation of the stack is maximized. Constraints (54) ensure that v equals the smallest payload violation over all item permutations. Due to (55), the payload violations are correctly computed. Finally, (56) guarantees that the weight variables w_i are contained in the given intervals.

Note that constraints (55) are non-linear, due to the product of α with w. To remove this non-linearity, we introduce new variables $\beta_{il}^k = w_i \alpha_l^k$ and require $0 \leq \beta_{il}^k \leq M\alpha_l^k$ and $w_i - M(1-\alpha_l^k) \leq \beta_{il}^k \leq w_i$ for a suitable large constant M (note that $M \geq \overline{w}_i$ suffices). This gives rise to the following new formulation:

$$\begin{aligned}
&\text{(Max-}w\text{)} \qquad \max \ v \\
&\text{s.t.} \sum_{l \in L'(q)} \sum_{h>l} \sum_{i \in I(q)} p_{ih}^k \beta_{il}^k - a \sum_{l \in L'(q)} \sum_{i \in I(q)} p_{il}^k \beta_{il}^k \geq v && \forall k \in \mathcal{K} \\
&\beta_{il}^k \leq \overline{w}_i \alpha_l^k && \forall k \in \mathcal{K}, l \in L'(q), i \in I(q) \\
&w_i + \overline{w}_i(\alpha_l^k - 1) \leq \beta_{il}^k \leq w_i && \forall k \in \mathcal{K}, l \in L'(q), i \in I(q) \\
&\underline{w}_i \leq w_i \leq \overline{w}_i && \forall i \in I(q) \\
&\alpha_l^k \in \{0,1\} && \forall k \in \mathcal{K}, l \in L'(q) \\
&\beta_{il}^k \geq 0 && \forall k \in \mathcal{K}, i \in I(q), l \in L'(q) \\
&v \geq 0
\end{aligned}$$

We denote this problem as (AV1, $I(q)$, $\mathcal{P}'$) and its optimal objective value by $\overline{v}_q(\mathcal{P}')$.

We now consider the second subproblem, which consists of finding a (possibly new) permutation for the items in stack q minimizing the total payload violation with respect to the current weights w, i.e., a solution of problem

$$\text{(Min-}\pi\text{)} \qquad \min_{\pi \in \mathcal{P}(I_q)} v_q(\pi, w).$$

For $i \in I(q), l \in L(q)$ we introduce binary variables p_{il} determining the item permutation, i.e., $p_{il} = 1$ if and only if item i is assigned to level l. Variables $v_l \geq 0$ for $l \in L'(q)$ are used to determine the payload violation at level l. Then, problem (Min-π) can be formulated as follows.

$$\text{(Min-}\pi\text{)} \quad \min \sum_{l \in L'(q)} v_l \tag{60}$$

$$\text{s.t.} \sum_{\substack{h \in L(q) \\ h > l}} \sum_{j \in I(q)} w_j p_{jh} - \sum_{i \in I(q)} a w_i p_{il} \leq v_l \qquad \forall l \in L'(q) \tag{61}$$

$$\sum_{i \in I(q)} p_{il} = 1 \qquad \forall l \in L(q) \tag{62}$$

$$\sum_{l \in L(q)} p_{il} = 1 \qquad \forall i \in I(q) \tag{63}$$

$$p_{i,l+1} + p_{jl} \leq 1 \qquad \forall i, j \in I(q) \text{ with } s_{ij} = 0, l \in L'(q) \tag{64}$$

$$p_{il} \in \{0, 1\} \qquad \forall i \in I(q), l \in L(q) \tag{65}$$

$$v_l \geq 0 \qquad \forall l \in L'(q) \tag{66}$$

Due to (60) the total payload violation is minimized. Constraints (61) are used to determine the violations v_l at the different levels. The assignment constraints (62) and (63) ensure that at each level exactly one item is stored and that each item is assigned to exactly one level, respectively. Due to constraints (64) the item permutation is feasible with respect to the stacking constraints s_{ij}. We denote this problem as (AV2, $I(q)$, w) and its optimal objective value by $v_q^*(w)$.

These two subproblems are then solved, until their objective values coincide and the worst possible total payload of a fixed stack assignment is determined, which also yields a new worst-case weight scenario. This scenario is added to the main adjustable problem, and the process is repeated. We summarize this approach in Algorithm 2.

Finally, we present a heuristic to solve (ARI), where we generate a set of candidate patterns (corresponding to subsets of items assigned to the same stack). To this end, we consider an extended problem formulation for (ARI). Let $\mathcal{C}$ denote all possible subsets $C \subseteq I$ of items that are feasible with respect to the stacking constraints and the stack capacity b (i.e., these items can be assigned together to the same stack). We introduce a binary variable y_C for each such subset C to decide if subset C is used. For each $C \in \mathcal{C}$, let

$$v(C) = \max_{w \in \mathcal{U}^I} \min_{\pi \in \mathcal{P}(C)} v(\pi, w)$$

be the worst-case payload violation of the set C (which can be determined using (AV1) and (AV2)).

This way, we have the following equivalent formulation (EARI) for (ARI):

$$\text{(EARI)} \quad \min \sum_{C \in \mathcal{C}} v(C) y_C \tag{67}$$

$$\text{s.t.} \sum_{C \in \mathcal{C}} \chi_j^C y_C = 1 \qquad \forall j \in I \tag{68}$$

$$\sum_{C \in \mathcal{C}} y_C \leq m \tag{69}$$

$$y_C \in \{0, 1\} \qquad \forall C \in \mathcal{C} \tag{70}$$

Here, χ_j^C indicates whether $j \in I$ is contained in C with $\chi_j^C = 1$ if and only if $j \in C$, and zero otherwise. Constraints (68) ensure that every item is contained in one set, and constraint (69) bounds the number of available subsets.

Algorithm 2 Exact Algorithm for (ARI, $\mathcal{U}$)

```
Require: An instance of (ARI, $\mathcal{U}$).
 1: $k \leftarrow 0$
 2: Take an arbitrary scenario $w^0 \in \mathcal{U}^I$ and let $\mathcal{U}^0 \leftarrow \{w^0\}$.
 3: Solve (ARF, $\mathcal{U}^k$). Let $x^k$ be the resulting stacking solution and $LB^k$ its objective value.
 4: $UB^k \leftarrow 0$
 5: for all $q \in Q$ do
 6:     $\ell \leftarrow 0$
 7:     Let $I(q)$ be the set of items assigned to stack $q$ in $x^k$.
 8:     $\mathcal{P}^0 \leftarrow \{\pi^0\}$ for some permutation $\pi^0$ of all items in $I(q)$ which is feasible w.r.t. $s_{ij}$.
 9:     Solve (AV1, $I(q)$, $\mathcal{P}^\ell$).
10:     Let $w$ be the resulting item weights and $\overline{v}_q(\mathcal{P}^\ell)$ the resulting objective value.
11:     Solve (AV2, $I(q)$, $w$).
12:     Let $\pi^\ell$ be the resulting item permutation, and $v_q^*(w)$ the resulting objective value.
13:     if $\overline{v}_q(\mathcal{P}^\ell) = v_q^*(w)$ then
14:         $UB^k \leftarrow UB^k + v_q^*(w)$
15:         $w_i^k \leftarrow w_i$ for all items $i \in I(q)$
16:     else
17:         $\mathcal{P}^{\ell+1} \leftarrow \mathcal{P}^\ell \cup \{\pi^\ell\}$
18:         $\ell \leftarrow \ell + 1$
19:         Goto 9
20:     end if
21: end for
22: if $UB^k = LB^k$ then
23:     return optimal stacking solution $x^k$
24: else
25:     $\mathcal{U}^{k+1} \leftarrow \mathcal{U}^k \cup \{w^k\}$
26:     $k \leftarrow k + 1$
27:     Goto 3
28: end if
```

As there are potentially exponentially many relevant sets $C \in \mathcal{C}$, we use the following algorithm to solve (EARI) heuristically: We generate a set $\mathcal{C}'$ of subsets $C \subseteq I$ at random, where the cardinality of each such subset C is randomly chosen between 1 and b. We evaluate each subset using (AV1) and (AV2) iteratively to compute $v(C)$. Then, we solve (EARI) using the heuristic set $\mathcal{C}'$ instead of the full set $\mathcal{C}$.

Note that problems of type (EARI) are easy to solve with current commercial MIP solvers (constraints (68) are special ordered set constraints, and constraint (69) is a single knapsack constraint), which means that a large number of sets C can be used in the heuristic with still small computation times. Thus, one can expect this approach to give better solutions than Algorithm 2 within the same amount of time; however, no quality bounds (or even a proof of optimality) are produced.

5. Computational experiments

5.1. Setup

To test the performance of the models and algorithms introduced in this paper, we performed four experiments with different sets of instances.

Recall that an uncertain stacking problem is parameterized by: The number of items n, the number of available stacks m, the maximum height of stacks b, and the payload violation parameter a. Additionally, a stacking matrix S is required as well as either an interval-based uncertainty $\mathcal{U}^I$ or a finite uncertainty set $\mathcal{U}^F$.

We randomly generated stacking matrices S by using a density parameter $d \in [0,1]$, which is the relative number of ones within the non-diagonal elements of the matrix (i.e., for $d = 1$, all items can be stacked onto each other, and for $d = 0.5$, there are $n(n-1)/2$ randomly distributed allowed pairings). Furthermore, we generated lower and upper bounds $\underline{w}_i$ and $\overline{w}_i$ on item weights in the following way: There are two types of items, which both occur with the same probability. The first type of items has $\underline{w}_i \in [9,10]$ and $\overline{w}_i \in [10,11]$; the second type of items has $\underline{w}_i \in [0,10]$ and $\overline{w}_i \in [10,20]$. Thus, the expected average of lower and upper bound is 10 in both cases, but the variance is different. This reflects the case that items have on average a similar weight, but different variance. For finite uncertainty sets, we sample scenarios uniformly.

For the four experiments, we modified different problem parameters:

- In the first experiment, we fix $a = 1$, $d = 0.5$ and $m = 3$, and vary the number of items n from 9 to 30 in steps of 3. The stack height b is equal to $n/3$, that is, this experiment considers relatively few but high stacks.
- In the second experiment, we fix $a = 1$, $d = 0.5$ and $b = 3$, and vary the number of items n from 9 to 30 in steps of 3. The number of stacks m is equal to $n/3$, that is, this experiment considers relatively many but small stacks.
- In the third experiment, we consider the impact of changing values for a. We choose $a = 0.5$ to $a = 1.5$ with a stepsize of 0.1. Other parameters are fixed to $n = 24$, $m = 4$, $b = 6$, $d = 0.5$.
- In the fourth and last experiment, different matrix densities d are considered. We generated instances with $d = 0.2$ to $d = 0.8$ with a stepsize of 0.1. As before, we use $n = 24$, $m = 4$, $b = 6$, and set $a = 1$.

Results on experiments 1 and 2 are described in the following, while results on experiments 3 and 4 can be found in the appendix. For each of the above parameter choices, we generated 20 instances (all of them were noted to be feasible). For each instance, we solve:

- The nominal model, where nominal weights are the midpoints of the respective intervals. We refer to this solution as "Nom".
- The strictly robust model with finite uncertainty, sampling 10 and 20 scenarios; we denote these solutions as "S-10" and "S-20", respectively.
- The strictly robust model with interval-based uncertainty using Algorithm 1. This is denoted as "SI".
- The strictly robust model with interval-based uncertainty using the compact model. This is denoted as "SIC".
- The adjustable model with finite uncertainty, sampling 5 and 10 scenarios; we denote these solutions as "A-5" and "A-10", respectively.

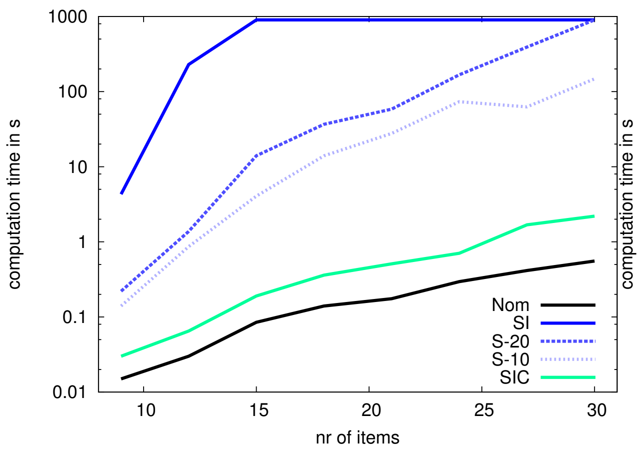
- The adjustable model with interval-based uncertainty using Algorithm 2. This is denoted as "AI".
- The adjustable model with interval-based uncertainty, based on the formulation (EARI), using the heuristic with 10,000 candidate sets. We denote this heuristic solution as "AIH".

Due to our parameter specificiation, every feasible solution must fill all available locations in the stacks. Therefore, we implemented AIH so that only item patterns of size b were generated; also, we ensured that all patterns were feasible by guiding the random item generation along the stacking matrix S.

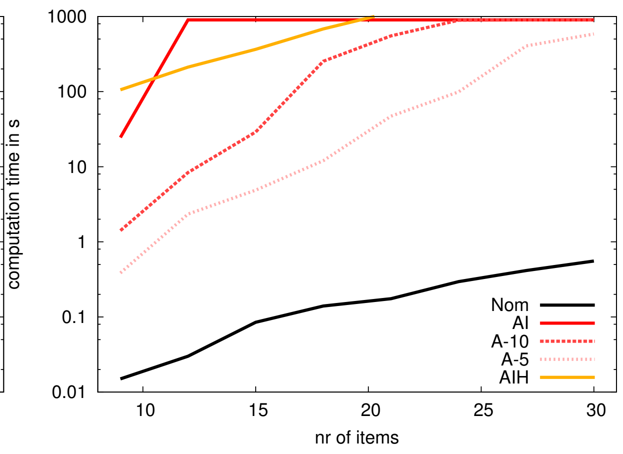
We used Cplex v.12.6 to solve all MIPs. All experiments were conducted on a computer with a 16-core Intel Xeon E5-2670 processor, running at 2.60 GHz with 20MB cache, and Ubuntu 12.04. Processes were pinned to one core. To restrict computation times, a time limit of 15 minutes was imposed on every solution approach except AIH.

5.2. Results, Experiment 1

The median computation times, depending on the number of items, are presented in Figure 1(a) (for the strict solutions) and Figure 1(b) (for the adjustable solutions). Note the logarithmic y-axis. The computation time of AIH mostly depends on the number of generated sets, and the time to evaluate them (which depends on b). Therefore, we can observe that the computation time of AIH increases with $n = 3b$ and surpasses the timelimit of 15 minutes from $n \geq 21$ on. Furthermore, nominal solution times are very small, while the iterative approaches hit the time limit of 15 minutes already for small n.



(a) Strict solutions.

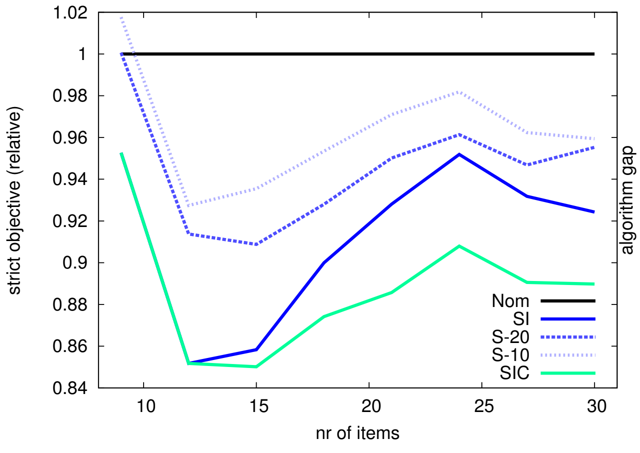


(b) Adjustable solutions.

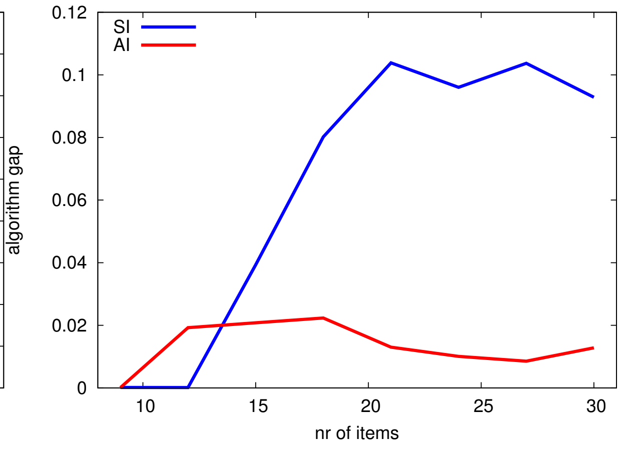
Figure 1: Results for experiment 1: Median computation times in seconds.

SIC shows excellent computation times in comparison to SI as well as S-10, and scales well with an increasing number of items.

We consider the strict objective values in more detail in Figure 2(a), and the adjustable objective values in Figures 3(a) and 3(b). Strict objective values are computed by evaluating the worst-case objective of a fixed stacking solution (i.e., by solving problem (V) for every stack), while adjustable objective values are computed by taking reordering within stacks into account (i.e., by solving (Max-w) for every stack). All objective values are normalized with respect to the objective value of the nominal solution; i.e., SIC has a strict objective value of around 85% of the strict objective value of the nominal solution for $n = 13$ and $n = 15$. For $n \geq 15$, the AI and

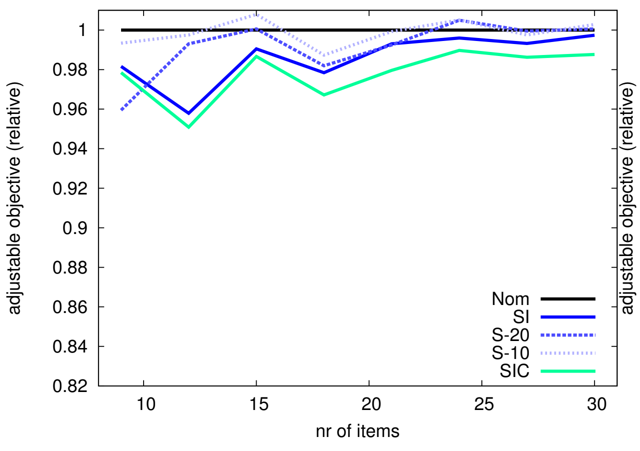


(a) Strict objective value.

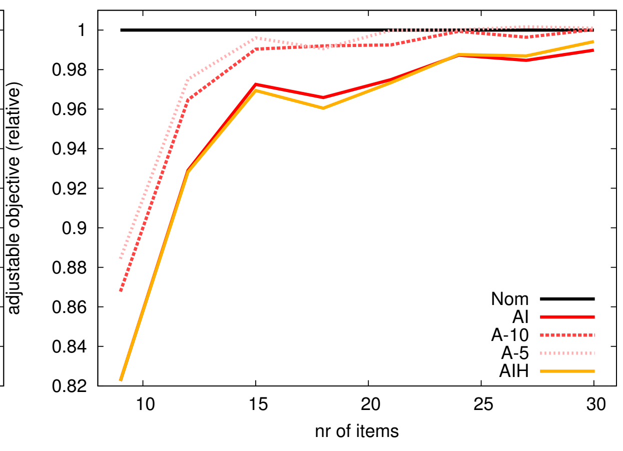


(b) Average algorithm gap.

Figure 2: Results for experiment 1: Strict objective value and algorithm gap.



(a) Strict solutions.



(b) Adjustable solutions.

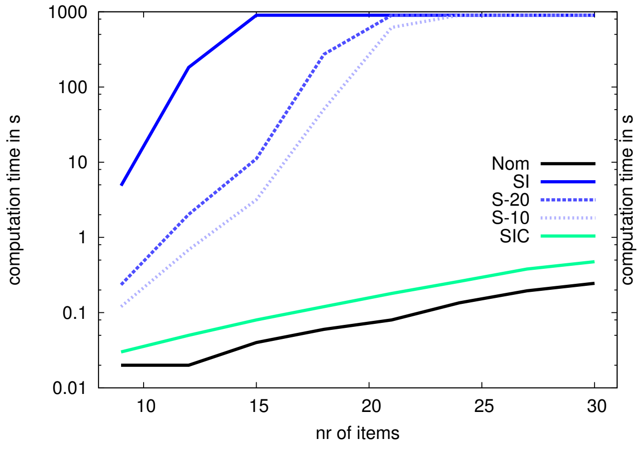
Figure 3: Results for experiment 1: Adjustable objective value.

AIH solutions (which perform best) barely improve upon the adjustable objective value of the nominal solution, and even the stricty robust solutions perform similarly. This can be explained with the fixed number of stacks and the increasing stack size, which gives more possibilities to rearrange items once the scenario becomes known. More generally speaking, for high stacks, a strict robust solution may have more potential than an adjustable robust solution.

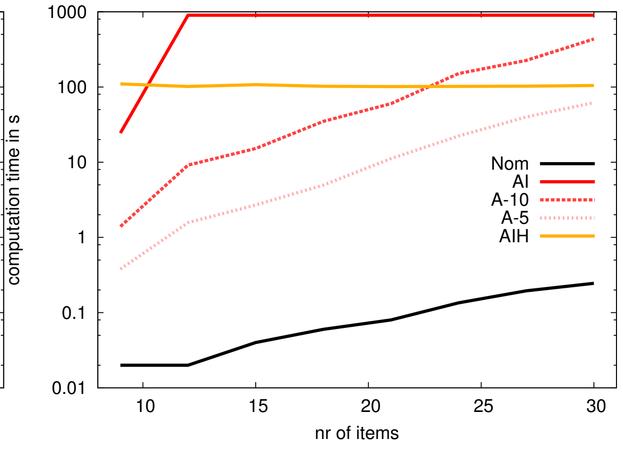
We analyze the quality of the SI and AI approaches in more detail in Figure 2(b), where the average algorithm gap is shown. The gap is derived from the ratio between the objective value of the current best solution (UB), and the lower bound (LB). As objective values are potentially equal to zero, we computed the gap as $1-(LB+1)/(UB+1)$. We find that adjustable objective values are within 2% of optimality and show a smaller gap than SI.

5.3. Results, Experiment 2

We now consider the case of fixed stack size and increasing number of stacks. In Figures 4–5 we show analogous plots as in Figures 1–2, comparing the median computation times, the relative strict and adjustable objective values, and the average algorithm gaps.

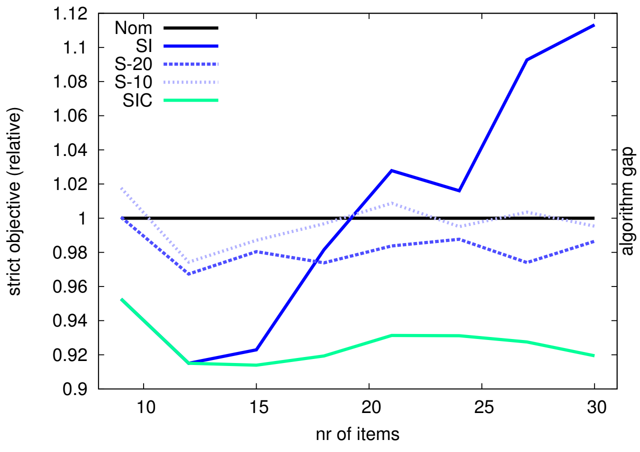


(a) Strict solutions.

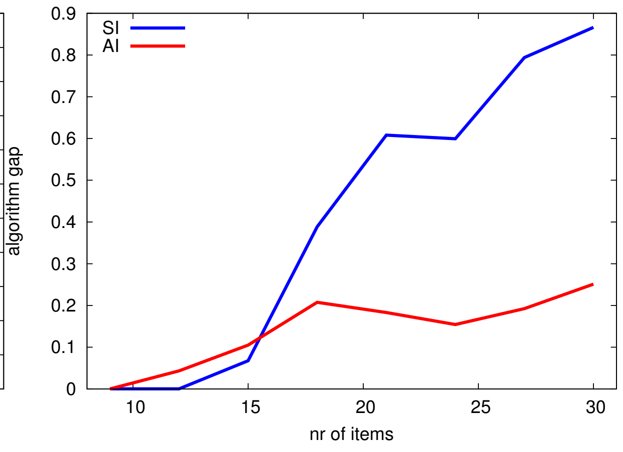


(b) Adjustable solutions.

Figure 4: Results for experiment 2: Median computation times in seconds.



(a) Strict objective value.

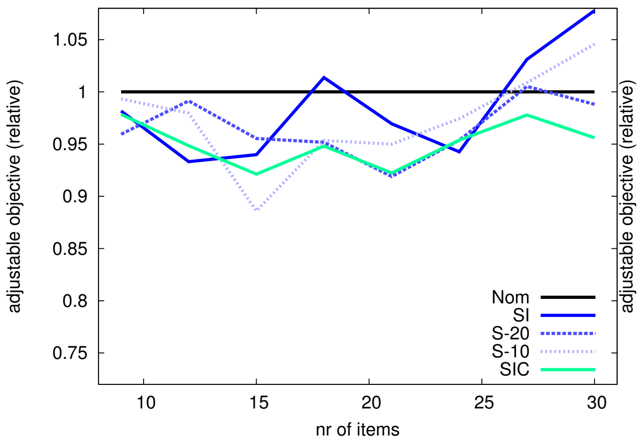


(b) Average algorithm gap.

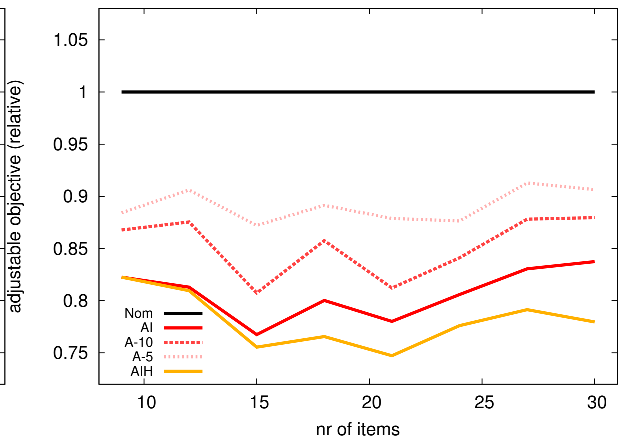
Figure 5: Results for experiment 2: Strict objective value and algorithm gap.

The two most notable differences in computation times are that the heuristic solution time does not increase with $n = 3m$ (see Figure 4(b)), as it is only depending on the stack size; and that the strict solutions take considerably more time to be solved, also giving a worse algorithm gap (see Figure 5(b)).

Regarding the objective values, we find that (in contrast to the first experiment), the strict objective values do not improve much upon the nominal solution (see Figure 5(a)), while the adjustable objective values show great potential for improvements (see Figure 6), with relative values between 75% and 80% for AIH (which performs best). Again more generally speaking, for small but many stacks, an adjustable robust solution may have more potential for a planner than a strict robust solution.



(a) Strict solutions.



(b) Adjustable solutions.

Figure 6: Results for experiment 2: Adjustable objective value.

6. Conclusions

We considered stacking problems with two kinds of constraints: The first is given by a general stacking matrix encoding which items can be stacked onto each other. This can be used to model practical requirements such as item departure times, or incompatible item dimensions. The second are payload constraints, which ensure that not more weight is stacked on top of an item than the stability of this item allows. However, as item weights are uncertain, we introduced robust stacking problems.

Two robust models were tackled: One where the complete item stacking needs to be fixed in advance before item weights are known; and one where adjustments in the item order can be made afterwards. Finite and interval-based uncertainty sets were considered and different solution approaches presented. In an extensive computational study on randomly generated instances, the impact of the number of items, the payload violation parameter, and the stacking matrix were analyzed.

We briefly review possible problem extensions in the following. Further uncertainty sets, such as interval-based uncertainty sets with additional restrictions may be considered. An example for such sets include the model of of Bertsimas and Sim (see [11]), where the total relative deviation of item weights from their nominal values is bounded by some parameter Γ. Using such an uncertainty set, Algorithms 1 and 2 are still applicable, with only slight differences in the computation of worst-case scenarios.

Finally, the adjustable approach presented in this paper can be extended by considering restrictions on the rearrangements that are allowed once the scenario becomes known. This is similar to the idea of recoverable robustness (see, e.g., [28]). We count the number of operations which are necessary to rearrange a stack. As an example, for a single stack, possible recovery cost measurements between two solutions x, x' include: The Hamming distance (i.e., the number of differently positioned items, given by $\sum_{i,l} |x_{iql} - x'_{iql}|$); or the number of items from top which have to be removed to transform x to x' or vice versa. In both cases, the recoverable robust counterpart for a finite number of scenarios can be modeled as a mixed-integer linear program similar to (ARF).

References

[1] H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

[2] D. Ambrosino, A. Sciomachen, and E. Tanfani. Stowing a containership: the master bay plan problem. *Transportation Research Part A*, 38:81–99, 2004.

[3] D. Ambrosino, A. Sciomachen, and E. Tanfani. A decomposition heuristics for the container ship stowage problem. *Journal of Heuristics*, 12:211–233, 2006.

[4] S. V. Amiouny, J. J. Bartholdi, J. H. Vande Vate, and J. Zhang. Balance loading. *Operations Research*, 40(2):238–246, 1992.

[5] A. Ben-Tal, B. D. Chung, S. R. Mandala, and T. Yao. Robust optimization for emergency logistics planning: Risk mitigation in humanitarian relief supply chains. *Transportation Research Part B: Methodological*, 45(8):1177–1189, 2011.

[6] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization.* Princeton University Press, Princeton and Oxford, 2009.

[7] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Math. Programming A*, 99:351–376, 2003.

[8] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[9] H. P. Benson. Concave minimization: Theory, applications and algorithms. In R. Horst and P. M. Pardalos, editors, *Handbook of Global Optimization*, volume 2 of *Nonconvex Optimization and Its Applications*, pages 43–148. Springer US, 1995.

[10] D. Bertsimas, D. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[11] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[12] P.C. Bouman, J.M. Akker, and J.A. van den Hoogeveen. Recoverable robustness by column generation. In *European Symposium on Algorithms*, volume 6942 of *Lecture Notes in Computer Science*, pages 215–226. Springer, 2011.

[13] F. Bruns, M. Goerigk, S. Knust, and A. Schöbel. Robust load planning of trains in intermodal transportation. *OR Spectrum*, 36:631–668, 2014.

[14] F. Bruns and S. Knust. Optimized load planning of trains in intermodal transportation. *OR Spectrum*, 34:511–533, 2012.

[15] F. Bruns, S. Knust, and N. Shakhlevich. Complexity results for storage loading problems with stacking constraints. *European Journal of Operational Research*, 2014. Submitted.

[16] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. Robust Algorithms and Price of Robustness in Shunting Problems. In *Proc. of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS07)*, 2007.

[17] A. L. Erera, J. C. Morales, and M. Savelsbergh. Robust optimization for empty repositioning problems. *Operations Research*, 57(2):468–483, 2009.

[18] M. Goerigk and B. Grün. A robust bus evacuation model with delayed scenario information. *OR Spectrum*, 36(4):923–948, 2014.

[19] M. Goerigk and A. Schöbel. Algorithm engineering in robust optimization. *LNCS State-of-the-Art Surveys Springer*, 2015. To appear.

[20] D. Halliday, R. Resnick, and J. Walker. *Principles of Physics, 10th Edition International Student Version.* John Wiley & Sons, 2014.

[21] J. Kang, K. R. Ryu, and K. H. Kim. Deriving stacking strategies for export containers with uncertain weight information. *Journal of Intelligent Manufacturing*, 17(4):399–410, 2006.

[22] K. H. Kim, Y. M. Park, and K. R. Ryu. Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, 124(1):89–101, 2000.

[23] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications.* Kluwer Academic Publishers, 1997.

[24] X.T. Le and S. Knust. MIP-based approaches for robust storage loading problems with stacking constraints. 2015. Technical report, University of Osnabrück.

[25] C.-Y. Lee and Q. Meng. *Handbook of ocean container transportation logistic: Making global supply chains effective.* Springer, 2015.

[26] J. Lehnfeld and S. Knust. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, 239(2):297–312, 2014.

[27] W. Li, Y. Wu, and M. Goh. *Planning and scheduling for maritime container yards: supporting and facilitating the global supply network.* Springer, 2015.

[28] C. Liebchen, M. Lübbecke, R. H. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R.H. Möhring, and C.D. Zaroliagis, editors, *Robust and online large-scale optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2009.

[29] K. Mathur. An integer-programming-based heuristic for the balanced loading problem. *Operations Research Letters*, 22:19–25, 1998.

[30] R. Montemanni. A Benders decomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 174(3):1479–1490, 2006.

[31] A. Sciomachen and E. Tanfani. The master bay plan problem: a solution method based on its connection to the three-dimensional bin packing problem. *IMA Journal of Management Mathematics*, 14:251–269, 2003.

[32] UNCTAD/RMT/2014. Review of maritime transport 2014. United Nations conference on Trade and Development, eISBN 978-92-1-056861-6.

[33] United Nations. Regional shipping and port development: Container traffic forecast 2007 update. United Nations: Economic and Social Comission for Asia and the Pacific (ESCAP), New York, 2007.

[34] W. Vancroonenburg, J. Verstichel, K. Tavernier, and G. V. Berghe. Automatic air cargo selection and weight balancing: A mixed integer programming approach. *Transportation Research Part E*, 65:70–83, 2014.

[35] B. Zeng and L. Zhao. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461, 2013.

[36] M. Zeng, M. Y. H. Low, W. J. Hsu, S. Y. Huang, F. Liu, and C. A. Win. Automated stowage planning for large containerships with improved safety and stability. In *Proceedings of the 2010 Winter Simulation Conference*, B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds., pages 1976–1989. Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ, 2010.

A. Relationship between the gravity center of a stack and the payload factor

In the following, we show in more detail how the gravity center of a stack is influenced by the payload factor a. We assume that we are given a single stack consisting of n items numbered by $1, \ldots, n$, where item i with weight w_i is stored at level i of the stack. Furthermore, the items have a common height $h > 0$ and the weight of each item is uniformly distributed over the item's volume. We are also given a fixed value of the payload parameter $a > 0$ and assume that the stacking configuration satisfies the payload constraints for each item in the stack. We compute the height h_C of the highest possible gravity center of the item set in the stack and prove that $h_C(a)$ is a monotonically increasing function. As a consequence, the smaller the value of a is, the lower $h_C(a)$ is, i.e., the more stable the stack is. Moreover, given a desired position for h_C, we can compute the payload parameter a corresponding to that position.

Since the stacking configuration satisfies the payload constraints for all items, we have

$$\sum_{i=0}^{k-1} w_{n-i} \leq a w_{n-k} \qquad (k = 1, \ldots, n-1).$$

Let us consider a general system consisting of n particles $i = 1, \ldots, n$ with weights w_i and assume that their centers of gravity are given by coordinates R_i (in d dimensions). Then, according to [20], Chapter 12 the coordinates G of the center of gravity of the whole system can be computed by

$$G = \frac{\sum_{i=1}^{n} w_i R_i}{\sum_{i=1}^{n} w_i}. \tag{71}$$

In the following, we consider only one dimension, namely the height of the stack.

It follows from (71) that the highest possible gravity center of the set consisting of the two topmost items n and $n-1$ is attained when $w_n = a w_{n-1}$. Similarly, the highest possible gravity center of the set consisting of the three topmost items $n, n-1, n-2$ is attained when

$$\begin{aligned} w_n &= a w_{n-1}, \\ w_n + w_{n-1} &= a w_{n-2}. \end{aligned}$$

By induction on n we can deduce that the highest possible gravity center of the item set in the stack is attained when

$$\begin{aligned} w_n &= a w_{n-1}, \\ w_n + w_{n-1} &= a w_{n-2}, \\ &\ldots \\ w_n + w_{n-1} + \ldots + w_{n-k+1} &= a w_{n-k}, \\ &\ldots \\ w_n + w_{n-1} + \ldots + w_2 &= a w_1, \end{aligned}$$

or equivalently,

$$\begin{aligned} w_{n-1} &= \frac{1}{a} w_n, \\ w_{n-k} &= \frac{(a+1)^{k-1}}{a^k} w_n \qquad (k = 2, \ldots, n-1). \end{aligned}$$

Therefore, we get

$$w_1 + \ldots + w_{n-2} + w_{n-1} + w_n = \left(\frac{(a+1)^{n-2}}{a^{n-1}} + \ldots + \frac{a+1}{a^2} + \frac{1}{a} + 1 \right) w_n = \frac{(a+1)^{n-1}}{a^n} w_n.$$

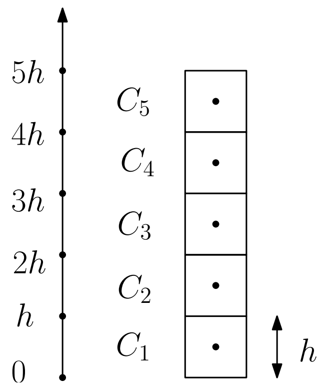


Figure 7: Stack with 5 items.

Since each item has a uniform distribution of its weight over its volume, the gravity center of each item is exactly in the middle of the item. More precisely, if we start with the height 0 from the ground level where all items are put on, and C_i denotes the position of the gravity center of item i, then the height h_{C_i} of C_i is $\left(i - \frac{1}{2}\right) h$ (see Figure 7), and we have

$$\sum_{i=1}^{n} w_i h_{C_i} = \sum_{i=1}^{n} \left(i - \frac{1}{2} \right) h w_i = \sum_{i=1}^{n} i w_i h - \frac{1}{2} h \sum_{i=1}^{n} w_i. \tag{72}$$

Let C be the gravity center of the items in the stack. Then by formula (71) the height of the highest possible possition of C can be computed by

$$\begin{aligned} h_C &= \frac{\sum_{i=1}^n w_i h_{C_i}}{\sum_{i=1}^n w_i} = \frac{\sum_{i=1}^n i w_i h - \frac{1}{2} h \sum_{i=1}^n w_i}{\sum_{i=1}^n w_i} = \frac{\sum_{i=1}^n i w_i h}{\sum_{i=1}^n w_i} - \frac{1}{2} h \\ &= \frac{\frac{(a+1)^{n-2}}{a^{n-1}} w_n h + \frac{(a+1)^{n-3}}{a^{n-2}} w_n 2h + \ldots + \frac{a+1}{a^2} w_n (n-2) h + \frac{1}{a} w_n (n-1) h + w_n n h}{\frac{(a+1)^{n-1}}{a^n} w_n} - \frac{1}{2} h \\ &= \frac{a}{a+1} h + 2 \frac{a^2}{(a+1)^2} h + \ldots + (n-2) \frac{a^{n-2}}{(a+1)^{n-2}} h + (n-1) \frac{a^{n-1}}{(a+1)^{n-1}} h + n \frac{a^n}{(a+1)^{n-1}} h - \frac{1}{2} h. \end{aligned}$$

By setting $u(a) := \frac{a}{a+1}$, we obtain

$$h_C = \left(u + 2u^2 + \ldots + (n-1) u^{n-1} \right) h + n a u^{n-1} h - \frac{1}{2} h =: g(a, u(a)) h.$$

Obviously, $u(a) = \frac{a}{a+1} = 1 - \frac{1}{a+1}$ is a monotonically increasing function with respect to a. Consequently, $g(a, u(a))$ is also monotonically increasing with respect to a. This means that $h_C(a)$ is a monotonically increasing function, or in particular, the smaller a is, the lower the highest possible gravity center h_C of the stack is.

B. SRI with $1 < a \leq 2$

In the following we present a compact formulation for (SRI) when $1 < a \leq 2$. This contains formulation (27)-(38) for $a < 1$, which is extended by new variables $w_{ql}^{k_1,k_2,k_3}$ representing the weight of the item in stack q at level l when there is a break at positions k_1 and k_3, as well as a light item atop a heavy item in level k_2. Additionally, variables $v_{ql}^{k_1,k_2,k_3}$ are used to measure the payload violation in this scenario. We denote $L^* := \{(k_1,k_2,k_3) \in (L \setminus \{b\}) \times (L \setminus \{b\}) \times (L \setminus \{b\}) : k_1 < k_2 < k_3\}$.

$$\min\ v \tag{73}$$

$$\text{s.t.} \quad \sum_{q \in Q} \sum_{l \in L} x_{iql} = 1 \qquad \forall i \in I \tag{74}$$

$$\sum_{i \in I} x_{iql} \leq 1 \qquad \forall q \in Q, l \in L \tag{75}$$

$$\sum_{j \in I \setminus \{i\}} s_{ij} x_{jq,l-1} \geq x_{iql} \qquad \forall i \in I, q \in Q, l \in L \setminus \{1\} \tag{76}$$

$$w_{ql}^{k} = \begin{cases} \sum_{i \in I} \underline{w}_i x_{iql} & \text{if } l \leq k \\ \sum_{i \in I} \overline{w}_i x_{iql} & \text{if } k < l \end{cases} \qquad \forall q \in Q, l \in L, k \in L \setminus \{b\} \tag{77}$$

$$w_{ql}^{k_1,k_2,k_3} = \begin{cases} \sum_{i \in I} \underline{w}_i x_{iql} & \text{if } l \leq k_1 \\ \sum_{i \in I} \overline{w}_i x_{iql} & \text{if } k_1 < l \leq k_2 \\ \sum_{i \in I} \underline{w}_i x_{iql} & \text{if } k_2 < l \leq k_3 \\ \sum_{i \in I} \overline{w}_i x_{iql} & \text{if } k_3 < l \end{cases} \qquad \forall q \in Q, l \in L, (k_1,k_2,k_3) \in L^* \tag{78}$$

$$\sum_{h > l} w_{qh}^{k} - a w_{ql}^{k} \leq v_{ql}^{k} \qquad \forall q \in Q, k, l \in L \setminus \{b\} \tag{79}$$

$$\sum_{h > l} w_{qh}^{k_1,k_2,k_3} - a w_{ql}^{k_1,k_2,k_3} \leq v_{ql}^{k_1,k_2,k_3} \qquad \forall q \in Q, l \in L, (k_1,k_2,k_3) \in L^* \tag{80}$$

$$\sum_{q \in Q} \sum_{l \in L \setminus \{b\}} v_{ql}^{k} \leq v \qquad \forall k \in L \setminus \{b\} \tag{81}$$

$$\sum_{q \in Q} \sum_{l \in L \setminus \{b\}} v_{ql}^{k_1,k_2,k_3} \leq v \qquad \forall (k_1,k_2,k_3) \in L^* \tag{82}$$

$$x_{iql} \in \{0,1\} \qquad \forall i \in I, q \in Q, l \in L \tag{83}$$

$$v_{ql}^{k} \geq 0 \qquad \forall q \in Q, l, k \in L \setminus \{b\} \tag{84}$$

$$v_{ql}^{k_1,k_2,k_3} \geq 0 \qquad \forall q \in Q, l, (k_1,k_2,k_3) \in L^* \tag{85}$$

$$w_{ql}^{k} \geq 0 \qquad \forall q \in Q, l \in L, k \in L \setminus \{b\} \tag{86}$$

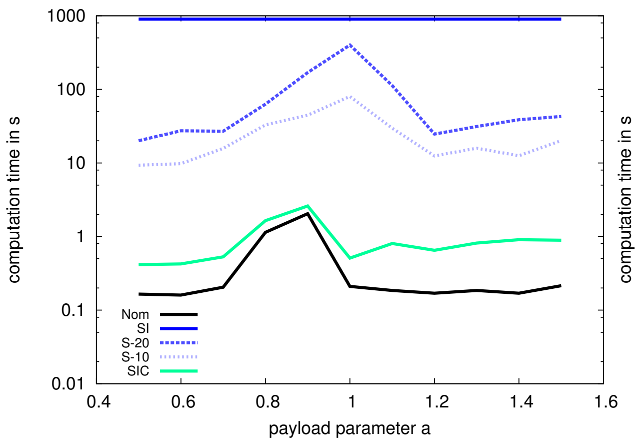
$$w_{ql}^{k_1,k_2,k_3} \geq 0 \qquad \forall q \in Q, l \in L, (k_1,k_2,k_3) \in L^* \tag{87}$$

$$v \geq 0 \tag{88}$$

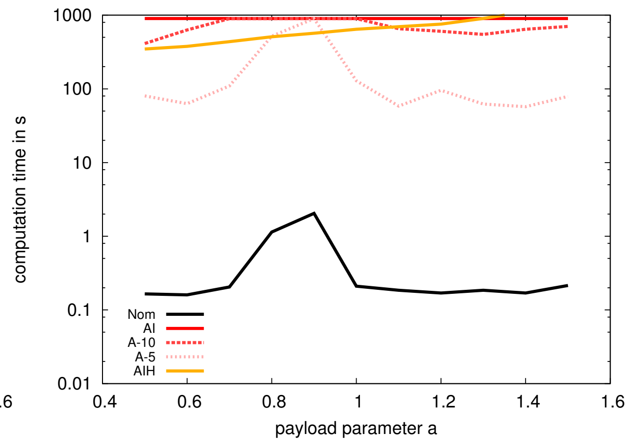
C. Results of Experiment 3

We now consider the impact of different values for the payload violation parameter a. Results are presented in Figures 8–9.

Note that computation times are not monotone in a (see Figure 8(a) and 8(b)), instead, problems with a being in the vicinity of 1 tend to take longer to solve for algorithms using a fixed number of sampled scenarios. Regarding the performance of SI, a considerable increase in its algorithm gap can be observed for $a \geq 1$ (see Figure 9(b)), along with loss in both strict and adjustable objective values (see Figures 9(a) and 10(a)). This behavior cannot be observed for S-10 or S-20, which indicates that the number of possible worst-case scenarios explodes for $a \geq 1$, rendering SI ineffective. This is not the case for SIC; even though the formulation size increases for $a > 1$, computation times are not strongly affected.

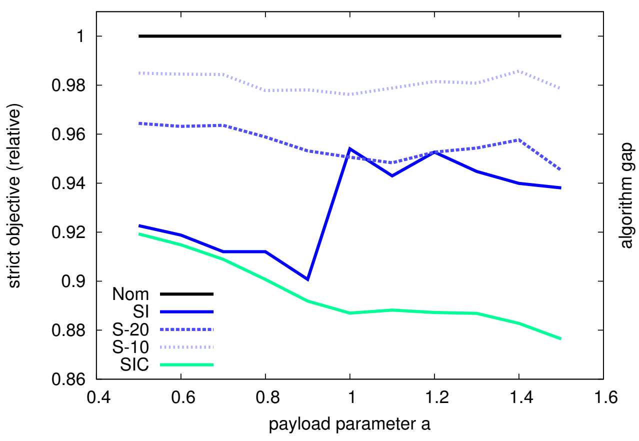


(a) Strict solutions.

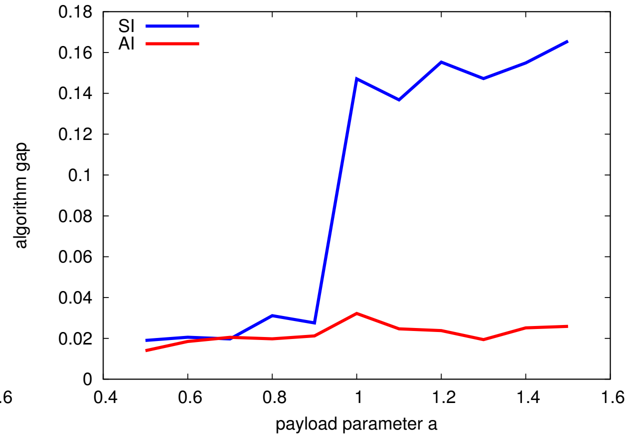


(b) Adjustable solutions.

Figure 8: Results for experiment 3: Median computation times in seconds.

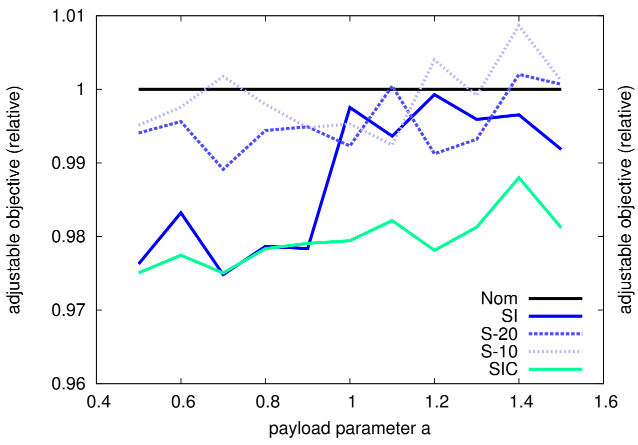


(a) Strict objective value.

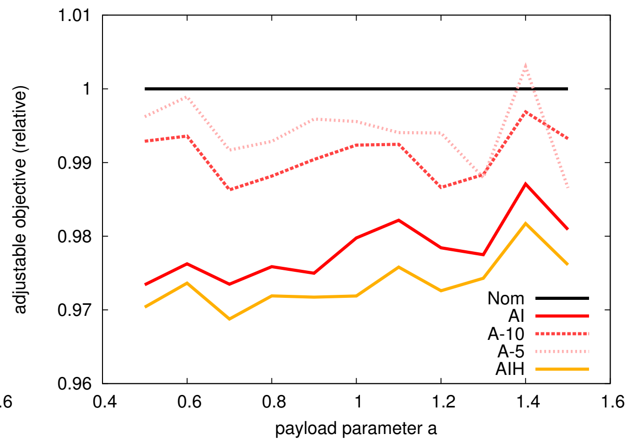


(b) Average algorithm gap.

Figure 9: Results for experiment 3: Strict objective value and algorithm gap.



(a) Strict solutions.

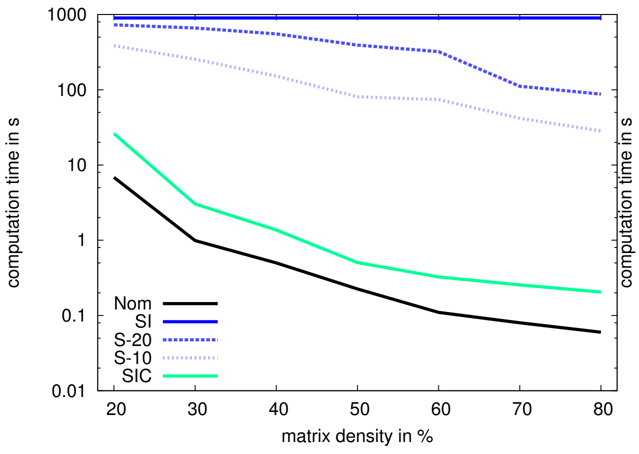


(b) Adjustable solutions.

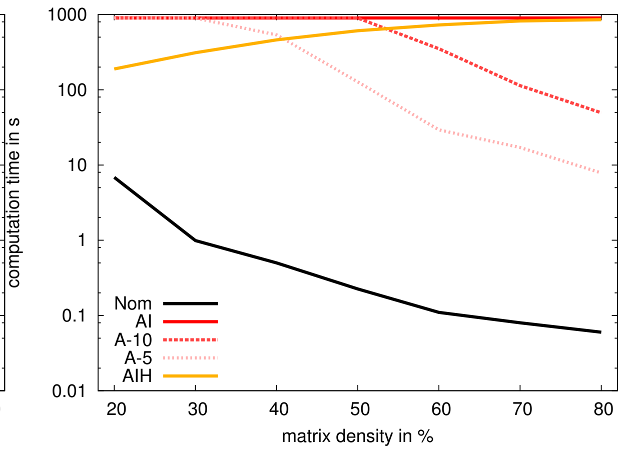
Figure 10: Results for experiment 3: Adjustable objective value.

D. Results of Experiment 4

In this final experiment, we evaluate the impact of the stacking matrix density d. Figures 11–12 visualize these results. An increasing matrix density tends to decrease computation times (except for AIH, where the evaluation process for a fixed set of items becomes more complex) as well as algorithm gaps, keeps the gain of using strict solutions constant, but decreases the gain in using adjustable solutions.

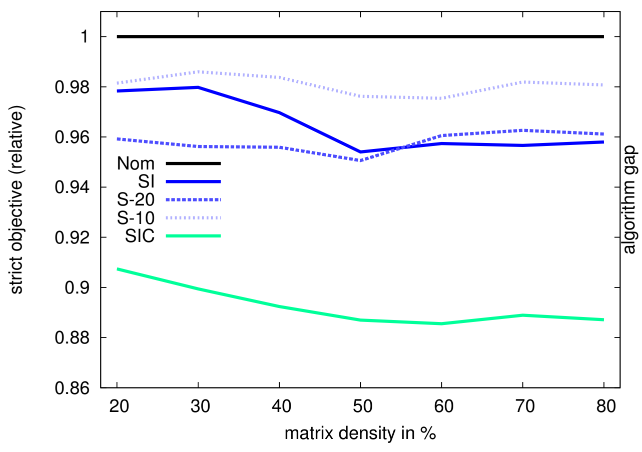


(a) Strict solutions.

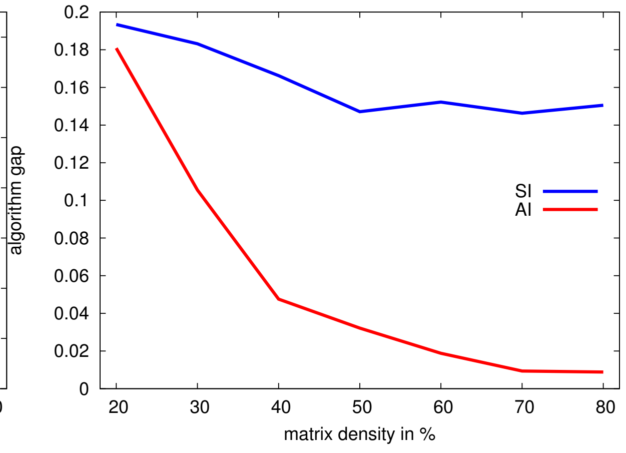


(b) Adjustable solutions.

Figure 11: Results for experiment 4: Median computation times in seconds.

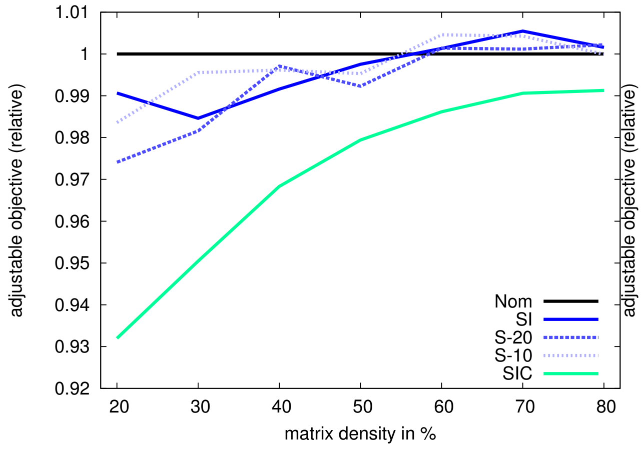


(a) Strict objective value.

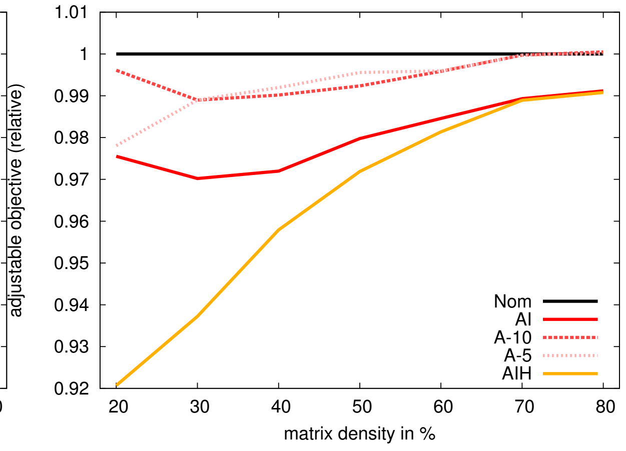


(b) Average algorithm gap.

Figure 12: Results for experiment 4: Strict objective value and algorithm gap.



(a) Strict solutions.



(b) Adjustable solutions.

Figure 13: Results for experiment 4: Adjustable objective value.