Balanced Dynamic Loading and Unloading[*]

Sándor P. Fekete[1], Sven von Höveling[1], Joseph S. B. Mitchell[†2], Christian Rieck[1], Christian Scheffer[1], Arne Schmidt[1], and James R. Zuber[2]

1 **Department of Computer Science, TU Braunschweig, Germany.**
`{s.fekete, v.sven, c.rieck, c.scheffer, arne.schmidt}@tu-bs.de`

2 **Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794, USA.**
`joseph.mitchell@stonybrook.edu, zuber139@gmail.com`

Abstract

We consider dynamic loading and unloading problems for heavy geometric objects. The challenge is to maintain balanced configurations at all times: minimize the maximal motion of the overall center of gravity. While this problem has been studied from an algorithmic point of view, previous work only focuses on balancing the *final* center of gravity; we give a variety of results for computing schemes that minimize the maximal motion of the center of gravity during the entire process.

In particular, we consider the one-dimensional case and distinguish between *loading* and *unloading*. In the unloading variant, the positions of the intervals are given, and we search for an optimal unloading order of the intervals. We prove that the unloading variant is NP-complete and give a 2.7-approximation algorithm. In the loading variant, we have to compute both the positions of the intervals and their loading order. We give optimal approaches for several variants that model different loading scenarios that may arise, e.g., in the loading of a container ship.

1 Introduction

Packing a set of objects is a classic challenge that has been studied extensively, from a variety of perspectives. The basic question is: how can the objects be arranged to fit into a container? Packing problems are important for many practical applications, such as loading items into a storage space, or containers onto a ship. They are also closely related to scheduling and sequencing, which may include additional temporal considerations. Packing and scheduling are closely intertwined in *loading* and *unloading* problems, where the challenge is not just to compute a good *final* configuration, but also to *dynamically build* this configuration, such that intermediate states are both achievable and stable, e.g., when loading and unloading container ships, for which maintaining *balance* throughout the process is crucial.

In this paper, we consider algorithmic problems of balanced loading and unloading. For unloading, this means planning an optimal sequence for removing a given set of objects, one at a time; for loading, this requires planning both the position and order of the objects.

The practical constraints of loading and unloading motivate a spectrum of relevant scenarios. As ships are symmetric around their main axis, we focus on one-dimensional settings, in which the objects correspond to intervals. Containers may be of uniform size, but stackable up to a certain limited height; because sliding objects on a moving ship are major safety hazards, stability considerations may prohibit gaps between containers.

[*] A full version of the paper is available at [2]. An extended abstract will appear in the 13th Latin American Theoretical INformatics Symposium (LATIN 2018), April 16–19, 2018.
[†] Work of this author is partially supported by the National Science Foundation (CCF-1526406).

1.1 Related Work

Previous work on cargo loading covers a wide range of specific aspects, constraints and objectives. The general Cargo Loading Problem (CLP) asks for an optimal packing of (possibly heterogeneous) rectangular boxes into a given bin, equivalent to the Cutting Stock Problem [4]. Most of the proposed methods are heuristics based on (mixed) integer programming and have been studied both for heterogeneous and homogeneous items.

Amiouny et al. [1] consider the problem of packing a set of one-dimensional boxes of different weights and different lengths into a flat bin (so they are not allowed to stack these boxes), in such a way that after placing the last box, the center of gravity is as close as possible to a fixed target point. They prove strong NP-completeness by a reduction from 3-Partition and give a heuristic with a guaranteed accuracy within $\ell_{max}/2$ of a given target point, where ℓ_{max} is the largest box length.

Gehring et al. [3] consider the general CLP, in which (rectangular) items may be stacked and placed in any possible position. Mongeau and Bes [5] consider a similar variant in which the objective is to maximize the loaded weight. In addition, there may be other paramaters, e.g., each item may have a different priority [8].

While all of this work is related to our problem, it differs in not requiring the center of gravity to be under control for each step of the loading or unloading process. A problem in which a constraint is imposed at each step of a process is Compact Vector Summation (CVS), which asks for a permutation of a set of k-dimensional vectors in order to control their sum, keeping each partial sum within a bounded k-dimensional ball. See Sevastianov [6, 7] for a summary of results in CVS and its application in job scheduling.

2 Preliminaries

An *item* is a unit interval $I := [m - \frac{1}{2}, m + \frac{1}{2}]$ with midpoint m. A set $\{I_1, \ldots, I_n\}$ of n items with midpoints $m_1, \ldots, m_n$ is *valid* if $m_i = m_j$ or $|m_i - m_j| \geq 1$ holds for all $i, j = 1, \ldots, n$. The *center of gravity* $C(I_1, \ldots, I_n)$ of a valid set $\{I_1, \ldots, I_n\}$ of items is defined as $\frac{1}{n}\sum_{i=1}^{n} m_i$.

Given a valid set $\{I_1, \ldots, I_n\}$ of items, we seek orderings in which each item I_j is removed or placed such that the maximal deviation for all points in time $j = 1, \ldots, n$ is minimized. Formally, for $j = 1, \ldots, n$ and a permutation $\pi : j \mapsto \pi_j$, let $C_j := C\left(I_{\pi_j}, \ldots, I_{\pi_n}\right)$.

The Unloading Problem (Unload) seeks to minimize the maximal deviation during an unloading process of $I_1, \ldots, I_n$. In particular, given an input set $\{I_1, \ldots, I_n\}$ of items, we seek a permutation π such that $\max_{i,j=1,\ldots,n} |C_i - C_j|$ is minimized.

In the Loading Problem (Load) we relax the constraint that the positions of the considered items are part of the input. In particular, we seek an ordering and a set of midpoints for the containers such that the containers are disjoint and the maximal deviation for all points in time of the loading process is minimized; see Section 4 for a formal definition.

3 Unloading

We show that the problem Unload is NP-complete and give a polynomial-time 2.7-approximation algorithm for Unload. We first show that there is a polynomial-time reduction from the discrete version of Unload, the Discrete Unloading Problem (dUnload), to Unload; this leads to a proof that Unload is NP-complete, followed by a 2.7-approximation algorithm for Unload.

In the Discrete Unloading Problem (dUnload), we consider a discrete set $X := \{x_1, \ldots, x_n\}$ of points. The center of gravity $C(X)$ of X is defined as $\frac{1}{n}\sum_{i=1}^{n} x_i$. For

$j = 1, \ldots, n$ and a permutation $\pi : j \mapsto \pi_j$, let $C_j = C\left(x_{\pi_1, \ldots, x_{\pi_j}}\right)$. Again, we seek a permutation such that $\max_{i,j=1,\ldots,n} |C_i - C_j|$ is minimized.

▶ **Lemma 3.1.** Unload *and* dUnload *are polynomial-time equivalent.*

3.1 NP-Completeness of the Discrete Case

We can establish NP-completeness of the discrete problem dUnload.

▶ **Theorem 3.2.** dUnload *is* NP*-complete.*

The proof of Theorem 3.3 is based on a reduction of 3-Partition and omitted for lack of space, just like any other formal proof; see the full version of this paper [2]. Because of the polynomial-time equivalence of dUnload and Unload, we conclude the following.

▶ **Corollary 3.3.** Unload *is* NP*-complete.*

3.2 Lower Bounds and an Approximation Algorithm

When unloading a set of items, their positions are fixed, so (after reversing time) unloading is equivalent to a loading problem with predetermined positions. For easier and uniform notation throughout the paper, we use this latter description.

In order to develop and prove an approximation algorithm for dUnload, we begin by examining lower bounds on the span, $R - L$, of a minimal interval, $[L, R]$, containing the centers of gravity at all stages in an optimal solution.

Without loss of generality, we assume that the input points x_i sum to 0 (i.e., $\sum_i x_i = 0$), so that the center of gravity, C_n, of all n input points is at the origin. We let $R = \max_i C_i$ and $L = \min_i C_i$. Our first simple lemma leads to a first (fairly weak) bound on the span.

▶ **Lemma 3.4.** *Let $(x_1, x_2, x_3, \ldots)$ be any sequence of real numbers, with $\sum_i x_i = 0$. Let $C_j = (\sum_{i=1}^{j} x_i)/j$ be the center of gravity of the first j numbers, and let $R = \max_i C_i$ and $L = \min_i C_i$. Then, $|R - L| \geq \frac{|x_i|}{i}$, for all $i = 1, 2, \ldots$.*

▶ **Corollary 3.5.** *For any valid solution to* dUnload*, the minimal interval $[L, R]$ containing the center of gravity at every stage must have length $|R - L| \geq \frac{|u_i|}{i}$ where u_i is the input point with the i-th smallest magnitude.*

We note that the naive lower bound given by Corollary 3.5 can be far from tight: Consider the sequence $1, 2, 3, 4, 5, 6, 7, -7, -7, -7, -7$. In the optimal order, the first -7 is placed fourth, after $2, 1, 3$. The optimal third and fourth centers, $\{2, -\frac{1}{4}\}$ are the largest magnitude positive and negative centers seen, and show a span 2.25 times greater than the naive bound of 1. By placing the first -7 in the third position, $R \geq \frac{3}{2}$, and $L \leq -\frac{4}{3}$. By placing it fifth, $R \geq \frac{5}{2}$. Our observation is that failing to place our first -7 if the cumulative sum is > 7 would needlessly increase the span.

This generalizes to the sequence $(x_1 = 1, x_2 = 2, \ldots, x_k = k, x_{k+1} = -k, x_{k+2} = -k, \ldots, x_N)$, with an appropriate x_N to make $\sum x_i = 0$. If we place positive weights in increasing order until, the current center of mass $C_j \geq \frac{k}{j}$, placing $-k$ instead of a positive point at position j would decrease the center of gravity well below $\frac{k}{j}$. The first negative point should be placed when $\min_j \frac{j^2 - j}{2} \geq k$, which is when $j \approx \sqrt{2k}$. In this example, our optimal center of gravity span is at least $\frac{k}{j} \approx \sqrt{\frac{k}{2}}$, not the 1 from the naive bound of Corollary 3.5.

We now describe our heuristic, $\mathcal{H}$, which leads to a provable approximation algorithm. It is convenient to relabel and reindex the input points as follows. Let $(P_1, P_2, \ldots)$ denote the positive input points, ordered (and indexed) by increasing value. Similarly, let $(N_1, N_2, \ldots)$ denote the negative input points, orders (and indexed) by increasing magnitude $|N_i|$ (i.e., ordered by decreasing value).

The heuristic $\mathcal{H}$ orders the input points as follows. The first point is simply the one closest to the origin (i.e., of smallest absolute value). Then, at each step of the algorithm, we select the next point in the order by examining three numbers: the partial sum, S, of all points placed in the sequence so far, the smallest magnitude point, α, not yet placed that has the same sign as S, and the smallest magnitude point, β, not yet placed that has the opposite sign of S. If $S+\alpha+\beta$ is of the same sign as S, then we place β next in the sequence; otherwise, if $S+\alpha+\beta$ has the opposite sign as S, then we place α next in the sequence. The intuition is that we seek to avoid the partial sum from drifting in one direction; we switch to the opposite sign sequence of input points in order to control the drift, when it becomes expedient to do so, measured by comparing the sign of S with the sign of $S+\alpha+\beta$, where α and β are the smallest magnitude points available in each of the two directions. We call the resulting ordering the $\mathcal{H}$-permutation. The $\mathcal{H}$-permutation puts the j-th largest positive point, P_j, in position π_j^+ in the order, and puts the j-th largest in magnitude negative point, N_j, in position π_j^- in the order, where

$$\pi_j^+ = j + \max_k\{k \ : \ \sum_{i=1}^{k} |N_i| \leq \sum_{l=1}^{j} P_l\} \quad \text{and} \quad \pi_j^- = j + \max_k\{k \ : \ \sum_{i=1}^{k} P_i < \sum_{l=1}^{j} |N_l|\}.$$

We obtain an improved lower bound based on our heuristic, $\mathcal{H}$, which orders the input points according to the $\mathcal{H}$-permutation.

▶ **Lemma 3.6.** *A lower bound on the optimal span of* dUnload *is given by* $|R-L| \geq \frac{P_i}{\pi_i^+}$ *and* $|R-L| \geq \frac{|N_i|}{\pi_i^-}$.

▶ **Claim 1.** *For any input set to the discrete unloading problem, where s_i is a member of S, the set of all terms with the same sign sorted by magnitude, a permutation π that minimizes the maximum value of the ratio $\frac{|s_i|}{\pi_i}$ must satisfy $\pi_k < \pi_i$, for all $k < i$.*

▶ **Theorem 3.7.** *The $\mathcal{H}$-permutation minimizes the maximum (over i) value of the ratio $\frac{|x_i|}{\pi_i}$, and thus yields a lower bound on $|R-L|$.*

For the worst-case ratio, we get the following.

▶ **Theorem 3.8.** *The $\mathcal{H}$ heuristic yields an ordering having span $R-L$ at most 2.7 times larger than the $\mathcal{H}$-lower bound.*

▶ **Corollary 3.9.** *There is a polynomial-time* 2.7*-approximation algorithm for* Unload.

4 Loading

We consider loading problems, where the positions of the objects are part of the optimization. Therefor some additional definitions are necessary:

An *item* is given by a real number ℓ. By assigning a *position* $m \in \mathbb{R}$ to an item, we obtain an interval I with length ℓ and midpoint m. For $n \geq 1$, we consider a set $\{\ell_1, \ldots, \ell_n\}$ of n items and assume $\ell_1 \geq \cdots \geq \ell_n$. Furthermore, $\{\ell_1, \ldots, \ell_n\}$ is *uniform* if $\ell := \ell_1 = \ldots = \ell_n$.

A *state* is a set $\{(I_1, h_1), \ldots, (I_n, h_n)\}$ of pairs, each one consisting of an interval I_j and an integer $h_j \geq 1$, the *layer* in which I_j lies. A state satisfies the following: (1) Two different intervals that lie in the same layer do not overlap and (2) for $j = 2, \ldots, n$, an interval in layer j is a subset of the union of the intervals in layer $j-1$.

A state $\{(I_1, h_1), \ldots, (I_n, h_n)\}$ is *plane* if all intervals lie in the first layer.

To simplify the following notations, we let m_j denote the midpoint of the interval I_j, for $j = \{1, \ldots, n\}$. The *center of gravity* $C(s)$ of a state $s = \{(I_1, h_1), \ldots, (I_n, h_n)\}$ is defined as $\frac{1}{M}\sum_{j=1}^{n} \ell_j m_j$, where M is defined as $\sum_{j=1}^{n} \ell_j$.

A *placement* p of n items $\ell_1, \ldots, \ell_n$ is a sequence $\langle I_{\pi_1}, \ldots, I_{\pi_n} \rangle$ such that (1) there is a permutation π with $\ell_i = |I_{\pi_i}|$ for all $i \in \{1, \ldots, n\}$ and (2) $\{(I_{\pi_1}, h_{\pi_1}), \ldots, (I_{\pi_j}, h_{\pi_j})\}$ is a state, the *j-th state* s_j, for each $j = 1, \ldots, n$. The 0-th state s_0 is defined as $\emptyset$ and its center of gravity $C(s_0)$ is defined as 0.

▶ **Definition 4.1.** The LOADING PROBLEM (LOAD) is defined as follows: Given a set of n items, determine a placement p such that the $n+1$ centers of gravity of the $n+1$ states of p lie close to 0. In particular, the *deviation* $\Delta(p)$ of a placement p is defined as $\max_{j=0,\ldots,n} |C(s_j)|$. We seek a placement of S with minimal deviation among all possible placements for S.

We say that *stacking is not allowed* if we require that all intervals are placed in layer 1. Otherwise, we say that *stacking is allowed.* For a given integer $\mu \geq 1$ we say that μ is the *maximum stackable height* if we require that all used layers are no larger than μ.

Note that in the loading case, minimizing the deviation is equivalent to minimizing the diameter, i.e., minimizing the maximal distance between the smallest and largest extent of the centers.

4.1 Optimally Loading Unit Items With Stacking

Now we consider the case where you have given a set of unit items which has to be loaded and you are allowed to stack these items up the a certain height. A simple and straightforward strategy for this scenario is to build a stack of maximum height first (call this stack $\mathcal{S}_0$) and place items as close as possible to $\mathcal{S}_0$ on alternating sides afterwards. By selecting the position of $\mathcal{S}_0$ carefully, this strategy guarantees the following:

▶ **Theorem 4.2.** *There is a polynomial-time algorithm for loading a set of unit items so that the deviation of the center of gravity is in* $[0, \frac{1}{1+\mu}]$, *where* μ *is the maximum stackable height.*

Furthermore it can be shown by a contradiction argument that there is no strategy that can guarantee a smaller deviation of the center of gravity than the strategy described above.

▶ **Theorem 4.3.** *The strategy given in Theorem 4.2 is optimal for* $n > \mu$, *i.e., there is no strategy such that the center of gravity deviates in* $[0, \frac{1}{1+\mu})$.

Combining Theorem 4.2 and Theorem 4.3 shows that our approach is optimal.

▶ **Corollary 4.4.** *With the given strategy for a uniform system where each item has length* ℓ, *the center of gravity deviates in* $[0, \frac{\ell}{1+\mu}]$, *which is optimal.*

4.2 Optimally Loading Without Stacking but With Minimal Space

Assume that the height of the ship to be loaded does not allow stacking items. This makes it necessary to ensure that the space consumption of the packing is minimal. We restrict ourselves to plane placements such that each state is connected. For simplicity, we assume w.l.o.g. that $\ell_1 \geq \cdots \geq \ell_n$ holds. First one can simply argue that $\Delta(p) \geq \frac{\ell_2}{4}$ holds for an arbitrary connected plane placement p of S. Subsequently we give an algorithm that realizes this lower bound.

A fundamental key for this subcase is that the center of gravity of a connected plane state is the midpoint of the induced overall interval.

▶ **Observation 1.** *Let s be a plane state such that the union of the corresponding intervals is an interval $[a, b] \subset \mathbb{R}$. Then $C(s) = \frac{a+b}{2}$.*

The algorithm works as follows. First, sort the items by decreasing value and place the item ℓ_1 at position $\frac{\ell_2}{4}$. After that, place all other items successively on alternating sides such that there are no gaps in the each intermediate placement. This approach yields a deviation of the center of gravity in $[-\frac{\ell_2}{4}, \frac{\ell_2}{4}]$.

▶ **Lemma 4.5.** *For each plane placement p of S, we have $\Delta(p) \geq \frac{\ell_2}{4}$.*

▶ **Lemma 4.6.** *We can compute a placement p of S such that $\Delta(p) \leq \frac{\ell_2}{4}$.*

The combination of Lemma 4.5 and Lemma 4.6 implies that our approach for connected placements is optimal.

▶ **Corollary 4.7.** *Given an arbitrary system, there is a polynomial-time algorithm for optimally loading a general set of items without stacking and under the constraint of minimal space consumption for all intermediate stages.*

5 Conclusion

We have introduced a new family of problems that seek to balance objects, controlling the variation of their center of gravity during the loading and unloading of the objects. We have provided hardness results and optimal or constant-factor approximation algorithms.

There are various related challenges. These include sequencing problems with multiple loading and unloading stops (which arise in vehicle routing or tour planning for container ships); variants in which items can be shifted in a continuous fashion; batch scenarios in which multiple items are loaded or unloaded at once (making it possible to maintain better balance, but also increasing the space of possible choices); and higher-dimensional variants, possibly with inhomogeneous space constraints. All these are left for future work.

References

1 Samir V. Amiouny, John J. Bartholdi, John H. Vande Vate, and Jixian Zhang. Balanced loading. *Operations Research*, 40(2):238–246, 1992.

2 S. P. Fekete, S. von Höveling, J. S. B. Mitchell, C. Rieck, C. Scheffer, A. Schmidt, and J. R. Zuber. Don't Rock the Boat: Algorithms for Balanced Dynamic Loading and Unloading. *CoRR*, abs/1712.06498, 2017. http://arxiv.org/abs/1712.06498.

3 Hermann Gehring, K Menschner, and M Meyer. A computer-based heuristic for packing pooled shipment containers. *Eur. J. Oper. Res.*, 44(2):277–288, 1990.

4 PC Gilmore and Ralph E Gomory. Multistage cutting stock problems of two and more dimensions. *Operations research*, 13(1):94–120, 1965.

5 Marcel Mongeau and Christian Bes. Optimization of aircraft container loading. *IEEE Transactions on Aerospace and Electronic Systems*, 39(1):140–150, 2003.

6 Sergey Sevastianov. On some geometric methods in scheduling theory: a survey. *Discrete Applied Mathematics*, 55(1):59–82, 1994.

7 Sergey Sevastianov. Nonstrict vector summation in multi-operation scheduling. *Annals of Operations Research*, 83:179–212, 1998.

8 Wouter Souffriau, Peter Demeester, G Vanden Berghe, and Patrick De Causmaecker. The aircraft weight and balance problem. *Proceedings of ORBEL*, 22:44–45, 1992.