

The Alternating Stock Size Problem and the Gasoline Puzzle

Alantha Newman^{*1}, Heiko Röglin^{†2}, and Johanna Seif^{‡3}

1 CNRS-Université Grenoble Alpes and G-SCOP, France

alantha.newman@grenoble-inp.fr

2 Department of Computer Science, University of Bonn, Bonn, Germany

roeglin@cs.uni-bonn.de

3 ENS Lyon, Lyon, France

johanna.seif@ens-lyon.fr

Abstract

Given a set S of integers whose sum is zero, consider the problem of finding a permutation of these integers such that: (i) all prefixes of the ordering are non-negative, and (ii) the maximum value of a prefix sum is minimized. Kellerer et al. referred to this problem as the *stock size problem* and showed that it can be approximated to within $3/2$. They also showed that an approximation ratio of 2 can be achieved via several simple algorithms.

We consider a related problem, which we call the *alternating stock size problem*, where the number of positive and negative integers in the input set S are equal. The problem is the same as above, but we are additionally required to alternate the positive and negative numbers in the output ordering. This problem also has several simple 2-approximations. We show that it can be approximated to within 1.79.

Then we show that this problem is closely related to an optimization version of the gasoline puzzle due to Lovász, in which we want to minimize the size of the gas tank necessary to go around the track. We present a 2-approximation for this problem, using a natural linear programming relaxation whose feasible solutions are doubly stochastic matrices. Our novel rounding algorithm is based on a transformation that yields another doubly stochastic matrix with special properties, from which we can extract a suitable permutation.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases approximation algorithms, stock size problem, scheduling with non-renewable resources

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.71

1 Introduction

Suppose there is a set of jobs that can be processed in any order. Each job requires a specified amount of a particular resource, e.g. gasoline, which can be supplied in an amount chosen from a specified set of quantities. The limitation is that the storage space for this resource is bounded, so it must be replenished as it is used. The goal is to order the jobs and the replenishment amounts so that the required quantity of the resource is always available for the job being processed and so that the storage space is never exceeded.

* Supported in part by LabEx PERSYVAL-Lab (ANR-11-LABX-0025).

† Supported by ERC Starting Grant 306465 (BeyondWorstCase).



More formally, we are given a set of integers $Z = \{z_1, z_2, \dots, z_n\}$ whose sum is zero. For a permutation σ , a prefix sum is $\sum_{i=1}^t z_{\sigma(i)}$ for $t \in [1, n]$. Our goal is to find a permutation of the elements in Z such that (i) each prefix sum is non-negative, and (ii) the maximum prefix sum is minimized. (Placing the elements with positive values in front of the elements with negative values satisfies (i) and therefore yields a feasible – although possibly far from optimal – solution.) This problem is known as the *stock size problem*. Kellerer, Kotov, Rendl and Woeginger presented a simple algorithm with a guarantee of $\mu_x + \mu_y$, where μ_x is the largest number in Z , and μ_y is the absolute value of the negative number with the largest absolute value in Z . (We sometimes use $\mu = \max\{\mu_x, \mu_y\}$.) Since both μ_x and μ_y are lower bounds on the value S^* of an optimal solution, this shows that the problem can be approximated to within a factor of 2. Additionally, they presented algorithms with approximation guarantees of $8/5$ and $3/2$ [11].

1.1 The Alternating Stock Size Problem

In this paper we first consider a restricted version of the stock size problem in which we require that the positive and negative numbers in the output permutation alternate. We refer to this problem as the *alternating stock size problem*. A motivation for this problem is that we could schedule tasks in advance of knowing the input data. For example, suppose we want to stock and remove items from a warehouse and each task occupies a certain time slot. If we want to plan ahead, we may want to designate each slot as a stocking or a removing slot in advance, e.g. all odd time slots will be used for stocking and all even time slots for de-stocking. This could be beneficial in situations where some preparation is required for each type of time slot.

The input for our new problem is two sets of positive integers, $X = \{x_1 \geq \dots \geq x_n\}$ and $Y = \{y_1 \geq \dots \geq y_n\}$, such that $|X| = |Y|$, and the two sets have equal sums. The elements of X represent the elements to be “added” and the elements of Y are those to be “removed”. Note that, here, $\mu_y = y_1$ and $\mu_x = x_1$. We now formally define the new problem.

► **Definition 1.** The goal of the *alternating stock size problem* is to find permutations σ and ν such that

- (i) for $t \in [1, n]$, $\sum_{i=1}^t x_{\sigma(i)} - y_{\nu(i)} \geq 0$,
- (ii) $\max_{1 \leq t \leq n} \sum_{i=1}^t (x_{\sigma(i)} - y_{\nu(i-1)})$ is minimized, where $y_{\nu(0)} = 0$.

Although this problem is a variant of the stock size problem, the algorithms found in [11] do not provide approximation guarantees since they do not necessarily produce feasible solutions for the alternating problem. Indeed, even the optimal solutions for these two problems on the same instance can differ greatly. The following example illustrates this.

$$\begin{aligned}
 X &= \{ \underbrace{p-1, \dots, p-1, 2}_{p \text{ entries}}, \underbrace{1, \dots, 1}_{p(p-1) \text{ entries}} \}, \\
 Y &= \{ \underbrace{p, \dots, p}_{p-1 \text{ entries}}, \underbrace{1, 1, 1, \dots, 1}_{p(p-1)+2 \text{ entries}} \}.
 \end{aligned}$$

For this instance, the optimal value for the alternating problem is at least $2p - 3$, while it is p for the original stock size problem. Thus, this example exhibits a gap arbitrarily close to 2 between the optimal solutions for the two problems.

We can show the following facts about the alternating problem. (i) There is always a feasible solution. (ii) The problem is NP-hard (as is the stock size problem). (iii) It is still the case that 2μ is an upper bound on the value of an optimal solution. Our main result for this problem is to give an algorithm with an approximation guarantee of 1.79 in Section 2.

1.2 Connections to the Gasoline Puzzle

The following well-known puzzle appears on page 31 in [12]:

Along a speed track there are some gas stations. The total amount of gasoline available in them is equal to what our car (which has a very large tank) needs for going around the track. Prove that there is a gas station such that if we start there with an empty tank, we shall be able to go around the track without running out of gasoline.

Suppose that the capacity of each gas station is represented by a positive integer and the distance of each road segment is represented by a negative integer. For simplicity, suppose that it takes one unit of gas to travel one unit of road. Then the assumption of the puzzle implies that the sum of the positive integers equals the absolute value of the sum of the negative integers. In fact, if we are allowed to permute the gas stations and the road segments (placing exactly one gas station between every pair of consecutive road segments), and our goal is to minimize the size of the gas tank required to go around the track (beginning from a feasible starting point), then this is exactly the alternating stock size problem.

This leads to the following natural problem: Suppose the road segments are fixed and we are only allowed to rearrange (i.e. permute) the gas stations. In other words, between each pair of consecutive road segments (represented by negative integers), there is a spot for exactly one gas station (represented by positive integers, the capacities), and we can choose which gas station to place in each spot. The goal is to minimize the size of the tank required to get around the track, assuming we can choose our starting gas station. What is the complexity of this problem?

We show in the full version of this paper that this problem is NP-hard [16]. Our algorithm for the alternating stock size problem specifically requires that there is flexibility in placing both the x -values and the y -values. Therefore, it does not appear to be applicable to this problem, where the y -values are pre-assigned to fixed positions. Let us now formally define the *gasoline problem*, which is the second problem we will consider in this paper.

1.3 The Gasoline Problem

As input, we are given the two sets of positive integers $X = \{x_1 \geq x_2 \geq \dots \geq x_n\}$ and $Y = \{y_1, y_2, \dots, y_n\}$, where the y_i 's are fixed in the given order and $\sum_{i=1}^n x_i = \sum_{i=1}^n y_i$. Our goal is to find a permutation π that minimizes the value of η :

$$\forall [k, \ell] : \left| \sum_{\pi(i) \in [k, \ell]} x_i - \sum_{i \in [k, \ell-1]} y_i \right| \leq \eta. \quad (1)$$

Given a circle with n points labeled 1 through n , the interval $[k, \ell]$ denotes a consecutive subset of integers assigned to points k through ℓ . For example, $[5, 8] = \{5, 6, 7, 8\}$, and $[n-1, 3] = \{n-1, n, 1, 2, 3\}$. We will often use μ_x to refer to x_1 , i.e. the maximum x -value, which is a lower bound on the optimal value of a solution.

Observe that in (1) we consider only intervals that contain one more x -value than y -value. One might argue that, in order to model our problem correctly, one also has to look at intervals that contain one more y -value than x -value. However, let I be such an interval and let $I' = [1, n] \setminus I$. Then the absolute value of the difference of the x -values and the y -values is the same in I and I' (with inverted signs) due to the assumption $\sum_{i=1}^n x_i = \sum_{i=1}^n y_i$.

We can also write the constraint (1) as:

$$\forall k : \sum_{\pi(i) \in [1, k]} x_i - \sum_{i \in [1, k-1]} y_i \leq \beta, \quad (2)$$

$$\forall k : \sum_{\pi(i) \in [1, k]} x_i - \sum_{i \in [1, k]} y_i \geq \alpha, \quad (3)$$

where $\alpha \leq 0$, $\beta \geq 0$ and $\eta = \beta - \alpha$. This version is slightly more general since it encompasses the scenario where we would like to minimize β for some fixed value of α . (With these constraints, it is no longer required that the sum of the x_i 's equals the sum of the y_i 's.)

What is the approximability of this problem? Getting a constant factor approximation appears to be a challenge since the following example shows that it is no longer the case that 2μ is an upper bound. Despite this, we show in Section 3 that there is in fact a 2-approximation algorithm for the gasoline problem.

Example showing unbounded gap between OPT and μ

Suppose X and Y each have the following n entries:

$$X = \underbrace{\{1, 1, \dots, 1, 1, 1, \dots, 1\}}_{n \text{ entries}}, \quad Y = \underbrace{\{2, 2, \dots, 2, 0, 0, \dots, 0\}}_{\substack{\frac{n}{2} \text{ entries} \\ \frac{n}{2} \text{ entries}}}.$$

In the example above, $\mu = 2$. However, the optimal value is $n/2$.

1.4 Generalizations of the Gasoline Problem

The requirement that the x - and y -jobs alternate may seem to be somewhat artificial or restrictive. A natural generalization of the gasoline problem (which we will refer to as the *generalized gasoline problem*) is where the y -jobs are assigned to a set of predetermined positions, which are not necessarily alternating. As in the gasoline problem, our goal is to assign the x -jobs to the remaining slots so as to minimize the difference between the maximum and the minimum prefix. There is a simple reduction from this seemingly more general problem to the gasoline problem. Let $X = \{x_1 \geq x_2 \geq \dots \geq x_{n_x}\}$ and $Y = \{y_1, y_2, \dots, y_{n_y}\}$ be the input, where the y -jobs are assigned to n_y (arbitrary) slots. The remaining n_x slots are for the x -jobs. To reduce to an instance of the gasoline problem (with alternation), we do the following. For each set of y -jobs assigned to adjacent slots, we add them up to form a single job in a single slot. For each pair of consecutive x -slots, we place a new y -slot between them where the assigned y -job has value zero. Thus, we obtain an instance of the gasoline problem as originally defined in the beginning of this section.

Our new algorithm, developed in Section 3 to solve the gasoline problem, can also be applied to a natural generalization of the alternating stock size problem, in which we relax the required alternation between the x - and y -jobs and consider a scenario in which each slot is labeled as an x - or a y -slot and can only accommodate a job of the designated type. In other words, in the solution, the x -jobs and y -jobs will follow some specified pattern that is not necessarily alternating. The goal is to find a feasible assignment of x - and y -jobs to x - and y -slots, respectively, that minimizes the difference between the prefixes with highest and lowest values. Since this is simply a generalization of the stock size problem with the additional condition that each slot is *slated* as an x - or a y -slot, we refer to this problem as the *slated stock size problem*.

Formally, we are given two sets of positive integers $X = \{x_1 \geq x_2 \geq \dots \geq x_{n_x}\}$ and $Y = \{y_1 \geq y_2 \geq \dots \geq y_{n_y}\}$, and $n = n_x + n_y$ slots, each designated as either an x -slot or a

y -slot. Let I_x and I_y denote the indices of the x - and y -slots, respectively, and let P denote a prefix. Then, the objective is to find a permutation π that minimizes the value of $\beta - \alpha$, where

$$\forall P: \quad \alpha \leq \sum_{\pi(i) \in P \cap I_x} x_i - \sum_{\pi(i) \in P \cap I_y} y_i \leq \beta. \quad (4)$$

For this problem, we obtain an algorithm with approximation guarantee $OPT + \mu_x + \mu_y$. The details for this analysis as well as all proofs not provided in this extended abstract can be found in the full version [16].

1.5 Related Work

The work most related to the alternating stock size problem is contained in the aforementioned paper by Kellerer et al. [11]. Earlier, Abdel-Wahab and Kameda studied a variant of the stock size problem in which the output sequence of the jobs is required to obey a given set of precedence constraints, but the stock size is also allowed to be negative. They gave a polynomial-time algorithm for the case when the precedence constraints are series parallel [1]. The gasoline problem and its generalization are related to those found in a widely-studied research area known as resource constrained scheduling, where the goal is usually to minimize the completion time or to maximize the number of jobs completed in a given timeframe while subject to some limited resources [4, 6]. For example, in addition to time on a machine, a job could require a certain amount of another resource and would be eligible to be scheduled only if the *inventory* for this resource is sufficient.

A general framework for these types of problems is called scheduling with non-renewable resources. Here, *non-renewable* means not abundantly available, but rather replenished according to some rules, such as periodically and in pre-determined increments (as in the gasoline problem), or in specified increments that can be scheduled by the user (as in the alternating stock size problem), or at some arbitrary fixed timepoints. Examples for scheduling problems in this framework are described by Briskorn et al., by Györgyi and Kis, and by Morsy and Pesch [5, 8, 9, 14]. While the admissibility of a schedule is affected by the availability of a resource (e.g. whether or not there is sufficient inventory), minimizing the inventory is not a main objective in these papers.

For example, suppose we are given a set of jobs to be scheduled on a single machine. Each job consumes some resource, and is only allowed to be scheduled at a timepoint if there is sufficient resource available for that job at this timepoint. Jobs may have different resource requirements. Periodically, at timepoints and in increments known in advance, the resource will be replenished. The goal is to minimize the completion time. If at some timepoint, there is insufficient inventory for any job to be scheduled, then no job can be run, leading to gaps in the schedule and ultimately a later completion time. This problem of minimizing the completion time is polynomial time solvable (sort the jobs according to resource requirement), but an optimal schedule may contain idle times.

Suppose that we have some investment amount α that we can add to the inventory in advance to ensure that there is always sufficient inventory to schedule some job, resulting in a schedule with no empty timeslots, i.e. the optimal completion time. There is a natural connection between this scenario and the gasoline problem: Let $|\alpha|$ in Equation (3) denote the available investment. For this investment, suppose we wish to minimize β , which is the maximum inventory, in order to complete the jobs in the optimal completion time. For any feasible α and β , our algorithm in Section 3 produces a schedule with the optimal completion time using inventory size at most $\beta + \mu$.

There are other works that directly address the problem of minimizing the maximum or cumulative inventory. Monma considers a problem in which each job has a specified effect on the inventory level [13]. Neumann and Schwindt consider a scheduling problem in which the inventory is subject to both upper and lower bounds [15]. However, to the best of our knowledge, our work is the first to give approximation algorithms for the problem of minimizing the maximum inventory for non-renewable resource scheduling with fixed replenishments.

The stock size problem is also closely related to the Steinitz problem, which is a well-known problem in discrepancy theory [2]. Given a set of vectors $v_1, v_2, \dots, v_n \in \mathbb{R}^d$ where $\|v_i\| \leq 1$ for some fixed norm and $\sum_{i=1}^n v_i = 0$, the Steinitz problem is to find a permutation of the vectors so that the norm of the sum of each prefix is bounded. There exists a permutation in which the norm of each prefix is at most d [7, 3]. It has been conjectured that this bound can be improved to $O(\sqrt{d})$, but only $O(\sqrt{d} \log^{2.5} n)$ is known [10]. The stock size problem is the one-dimensional analogue of the Steinitz problem. The variants of the stock size problem that we introduce in this paper can be extended to higher dimensions.

2 Algorithms for the Alternating Stock Size Problem

The existence of a feasible solution for the alternating stock size problem follows from the solution for the gasoline puzzle. Furthermore, the upper bound of 2μ is also tight for the alternating problem. If we modify the example given in [11], we have an example for the alternating problem with an optimal stock size of $2p - 3$, while $\mu = p$.

$$X = \{ \underbrace{p-1, \dots, p-1}_{p \text{ entries}}, 2 \}, \quad Y = \{ \underbrace{p, \dots, p}_{p-1 \text{ entries}}, 1, 1 \}.$$

In this section, we will present algorithms for the alternating stock size problem. We will use the notion of a (q, T) -pair, which is a special case of a (q, T) -batch introduced and used by [11] for the stock size problem.

► **Definition 2.** [11] A pair of jobs $\{x, y\}$, for $x \in X$ and $y \in Y$, is called a (q, T) -pair for positive reals T and $q \leq 1$, if:

- (i) $x, y \leq T$, (ii) $|x - y| \leq qT$.

The following lemma is a special case of Lemma 3 in [11], and the proofs are identical.

► **Lemma 3.** For positive T , $q \leq 1$ and a set of jobs partitioned into (q, T) -pairs, we can find an alternating sequence of the jobs with maximum stock size less than $(1 + q)T$.

2.1 The Pairing Algorithm

We now consider the simple algorithm that pairs x - and y -jobs, and then applies Lemma 3 to sequence the pairs. Suppose that there is some specific pairing that matches each x_i to some y_j , and consider the difference $x_i - y_j$ for each pair. Let $\alpha_1 \geq \dots \geq \alpha_{n_1}$ denote the positive differences, and let $\beta_1 \geq \dots \geq \beta_{n_2}$ denote the absolute values of the negative differences, where $n_1 + n_2 = n$.

► **Lemma 4.** The matching M^* that matches x_i and y_i for all $i \in \{1, \dots, n\}$ minimizes both α_1 and β_1 .

The pairing given by M^* directly results in a 2-approximation for the alternating stock size problem, by applying Lemma 3. Without loss of generality, let us assume that $\max\{\alpha_1, \beta_1\} =$

α_1 , and observe that $\alpha_1 \leq \mu$. Then M^* partitions the input into $(\alpha_1/\mu, \mu)$ -pairs. Applying Lemma 3, we obtain an algorithm that computes a solution with value at most $\mu + \alpha_1 \leq 2\mu$. We note that if $\alpha_1 \leq (1 - \epsilon)\mu$, then we have a $(2 - \epsilon)$ -approximation.

2.2 Lower Bound for the Alternating Stock Size Problem

In order to obtain an approximation ratio better than 2, we need to use a lower bound that is more accurate than μ . We now introduce a lower bound closely related to the one given for the stock size problem in [11] (Lemma 8). We refer to a real number C , which divides the sets X and Y into sets of small jobs and big jobs, as a *barrier*. Let $C \leq \mu$ be a barrier such that:

$$X = \{a_1 \geq a_2 \geq \dots \geq a_{n_a} \geq C > v_k \geq v_{k-1} \geq \dots \geq v_1\}, \quad (5)$$

$$Y = \{b_1 \geq b_2 \geq \dots \geq b_{n_b} \geq C > w'_1 \geq w'_2 \geq \dots w'_{n_a-n_b} \geq w_1 \geq w_2 \geq \dots \geq w_k\}, \quad (6)$$

where, without loss of generality, $n_a \geq n_b$. (If not then by swapping the x 's and the y 's we have a symmetric sequencing problem with $n_a \leq n_b$). The elements of (5) are all the x -jobs (partitioned into the sets A and V) and the elements of (6) are all the y -jobs. The jobs in Y that have value at most C are partitioned into W' and W .

Let $A' = \{a_{n_b+1}, \dots, a_{n_a}\} = \{a'_1, \dots, a'_{n_a-n_b}\}$, let V_i denote the i smallest v_j 's, i.e. $\{v_1, v_2, \dots, v_i\}$, and let W_i denote the i largest w_j 's in W , i.e. $\{w_1, w_2, \dots, w_i\}$. (Note that A' , V_i , and W_i each depend on C , but in order to avoid cumbersome notation, we do not use superscript C .) Let $s \in \{1, \dots, n_a - n_b\}$. After fixing a barrier C , let h be the (unique) index such that $w_h > v_h$ and $w_{h+1} \leq v_{h+1}$, and recall that S^* is the value of an optimal ordering. Then we obtain the following lower bound on S^* .

► **Lemma 5.** *For $n_a > n_b$, $1 \leq s \leq n_a - n_b$, the following inequality holds:*

$$S^* \geq LB(C) = \frac{1}{n_a - n_b - s + 1} \left(2 \sum_{i=s}^{n_a-n_b} a'_i - \sum_{i=s}^{n_a-n_b} w'_i + \sum_{i=1}^h (v_i - w_i) \right).$$

2.3 Alternating Batches: Definition

We need a few more tools before we can outline our new algorithm. The notion of *batches* introduced in [11], to which we briefly alluded before Lemma 3, is quite useful for the stock size problem. For $B \subseteq X \cup Y$, let $x(B)$ and $y(B)$ denote the total value of the x -jobs and y -jobs, respectively, in B . In its original form, the batching lemma (Lemma 3, [11]) calls for a partition of the input into groups or batches such that for some fixed positive real numbers T and $q \leq 1$, each group B has the following properties: $x(B), y(B) \leq T$ and $|x(B) - y(B)| \leq qT$. Given such a partition of the input, a sequence with stock size at most $(1 + q)T$ can be produced.

This approach is not directly applicable to the alternating stock size problem, because the output is not necessarily an alternating sequence. However, we will now show that the procedure can be modified to yield a valid ordering. With this goal in mind, we define a new type of batch, which we call an *alternating batch*. An alternating batch will either contain two jobs (small) or more than two jobs (large).

The modified procedure to construct an ordering of the jobs first partitions the input into alternating batches, then orders these batches, and finally orders the jobs contained within each batch. In the case of a small alternating batch, the batch will contain both an x -job and a y -job, and the last step simply preserves this order. A large alternating batch will be

required to fulfill certain additional properties that allow the elements to be sequenced in a way that is both alternating and feasible, i.e. all prefixes are nonnegative.

Suppose $B = \{(x'_1, y'_1), (x'_2, y'_2), \dots, (x'_\ell, y'_\ell)\}$, and consider the following four properties:

- (i) $\sum_{i=1}^{\ell} x'_i - \sum_{i=1}^{\ell} y'_i \geq 0$,
- (ii) $x'_1 - y'_1 \geq 0$,
- (iii) $x'_i - y'_i \leq 0$, for $2 \leq i \leq \ell$,
- (iv) $y'_1 \geq y'_2 \geq \dots \geq y'_\ell$.

► **Lemma 6.** *If a batch B satisfies properties (i), (ii), (iii) and (iv), then we can sequence the elements in B so that the items alternate, each prefix is non-negative and the maximum height (or prefix sum) of the sequence is x'_1 .*

► **Definition 7.** We call a set B a $(1-\epsilon)$ -alternating batch if $B = \{(x'_1, y'_1), (x'_2, y'_2), \dots, (x'_\ell, y'_\ell)\}$ such that:

- (1) $|\sum_{i=1}^{\ell} x'_i - \sum_{i=1}^{\ell} y'_i| \leq (1-\epsilon)\mu$,
- (2) if $\ell > 1$, then conditions (i) to (iv) hold.

► **Definition 8.** We say that a $(1-\epsilon)$ -alternating batch with more than two jobs is a *large* alternating batch. In other words, a large alternating batch obeys conditions (1) and (2) in Definition 7. A *small* alternating batch contains only two jobs and obeys condition (1) in Definition 7.

Note that, by definition, in a large alternating batch B , the sum of the x -jobs in B is at least the sum of the y -jobs in B .

► **Lemma 9.** *If the sets X and Y can be partitioned into large and small $(1-\epsilon)$ -alternating batches, then we can find an alternating sequence with maximum stock size less than $(2-\epsilon)\mu$.*

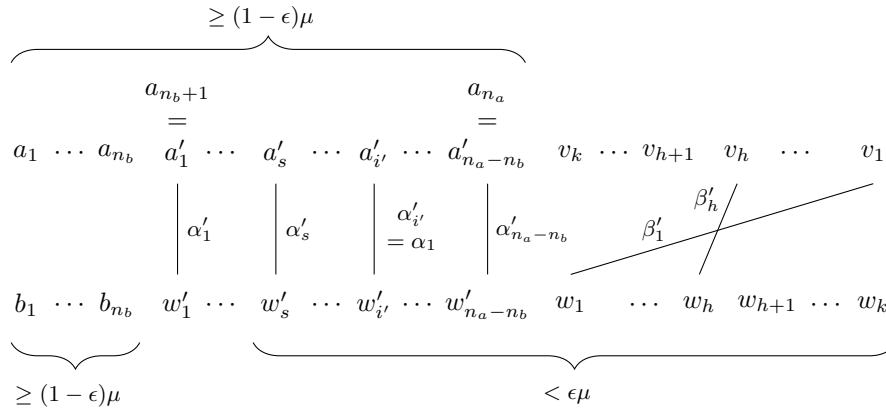
2.4 Alternating Batches: Construction

In this section, we present the final tool required for our algorithm. Suppose that for some $\epsilon : 0 \leq \epsilon \leq 1$, the following conditions hold for an input instance to the alternating stock size problem: $\alpha_1 > (1-\epsilon)\mu$, and $LB(C) < \frac{2}{2-\epsilon}\mu$, for $C = (1-\epsilon)\mu$. Then, we claim, there is some value of ϵ (to be determined later) for which the above two conditions can be used to partition the input into $(1-\epsilon)$ -alternating batches, to which we can then apply Lemma 9. In this section, we will heavily rely on the notation introduced in Section 2.2.

The sets $A' = \{a'_1, \dots, a'_{n_a - n_b}\}$ and $W' = \{w'_1, \dots, w'_{n_a - n_b}\}$ contain exactly the pairs in M^* that are split by barrier C . Let s be the smallest index such that $w'_s < \epsilon\mu$. To see that such an s actually exists, we note the following. Let i^* denote the index such that $x_{i^*} - y_{i^*} = \alpha_1$. Then $y_{i^*} < \epsilon\mu$ and the pair (x_{i^*}, y_{i^*}) is split by C . Thus, y_{i^*} corresponds to some $w'_{i'}$, and therefore $s \leq i'$. See Figure 1 for a schematic drawing.

For i in $\{1, \dots, n_a - n_b\}$, we define $\alpha'_i = a'_i - w'_i$ and for j in $\{1, \dots, h\}$, $\beta'_j = w_j - v_j$. (Recall that for $j \in \{1, \dots, h\}$, $w_j - v_j > 0$.) Furthermore, let \mathcal{A}_i denote the pair $\{a'_i, w'_i\}$ and let \mathcal{B}_j denote the pair $\{v_j, w_j\}$. Since $w'_s < \epsilon\mu$, it follows that all w'_i 's in W also have value less than $\epsilon\mu$. Moreover, $\beta'_j < \epsilon\mu$ for $j \in \{1, \dots, h\}$.

Our goal is now to construct $(1-\epsilon)$ -alternating batches. For each $i \in \{1, \dots, s-1\}$, note that $\alpha'_i \leq (1-\epsilon)\mu$. The set \mathcal{A}_i therefore forms a small $(1-\epsilon)$ -alternating batch. For each \mathcal{A}_i where $i \in \{s, \dots, n_a - n_b\}$, we will find a set of \mathcal{B}_j 's that can be grouped with this \mathcal{A}_i to create a large $(1-\epsilon)$ -alternating batch. However, to do this, we require that the condition on ϵ found in Claim 10 be satisfied.



■ **Figure 1** An illustration of the various elements used in the construction of the lower bound.

► **Claim 10.** *The condition*

$$2(1 - \epsilon) - \frac{2}{2 - \epsilon} > 2\epsilon \tag{7}$$

is satisfied when $\epsilon = .21$.

► **Lemma 11.** *If $LB(C) < 2\mu/(2 - \epsilon)$, $C = (1 - \epsilon)\mu$, and $2(1 - \epsilon) - \frac{2}{2 - \epsilon} > 2\epsilon$, then $\sum_{i=1}^h \beta'_i + \sum_{i=s}^{n_a-n_b} w'_i > 2\epsilon\mu(n_a - n_b - s + 1)$.*

For ease of notation, we set $d = n_a - n_b - s + 1$. In the following lemma, we show that we can also construct a $(1 - \epsilon)$ -alternating batch for each \mathcal{A}_i for $i \in [s, n_a - n_b]$.

► **Lemma 12.** *There exists d disjoint subsets S_1, \dots, S_d of $\{\mathcal{B}_1, \dots, \mathcal{B}_h\}$ such that for all i in $\{1, \dots, d\}$, the set $S_i \cup \mathcal{A}_{i+s-1}$ is a $(1 - \epsilon)$ -alternating batch.*

Now we want to complete the construction of the $(1 - \epsilon)$ -alternating batches, so that we can apply Lemma 9. For the sets \mathcal{A}_i , where $i \in \{s, \dots, n_a - n_b\}$, we construct batches according to Lemma 12. Let $y_{i^*} = w'_s$. For all $i < i^*$, the pair (x_i, y_i) form a small $(1 - \epsilon)$ -alternating batch. This follows from the fact that for all $i < i^*$, $y_{i^*} \geq \epsilon\mu$, by definition of s . Finally, if there are remaining elements, they are v_i 's and w_i 's, which can be paired up arbitrarily to construct more small $(1 - \epsilon)$ -alternating batches, since each remaining v_i has value strictly less than $(1 - \epsilon)\mu$ due to our choice of barrier, and each remaining w_i has value at most $\epsilon\mu$. Since the only limits on the value of ϵ are imposed by Lemma 11, we can set $\epsilon = .21$ and partition the input into .79-alternating batches.

2.5 A 1.79-Approximation Algorithm

We are now ready to present an algorithm for the alternating stock size problem with an approximation guarantee of 1.79.

► **Theorem 13.** *Algorithm 1 is a 1.79-approximation for the alternating stock size problem.*

3 Gasoline Problem

Let the variable z_{ij} be 1 if gas station x_i is placed in position j , and be 0 otherwise. Then we can formulate the gasoline problem as the following integer linear program whose solution matrix Z is a permutation matrix.

Algorithm 1 1.79-approximation

- 1: **Input:** the sets X and Y of positive numbers sorted in nonincreasing order.
 - 2: **Output:** a sequence that is a 1.79-approximation.
 - 3: Set $\epsilon = .21, C = (1 - \epsilon)\mu$.
 - 4: Match each x_i with y_i .
 - 5: **if** $\alpha_1 \leq (1 - \epsilon)\mu$ or if $LB(C) \geq \frac{2}{2-\epsilon}\mu$ **then**
 - 6: **return** solution for the Pairing Algorithm with guarantee of most $\mu + \alpha_1$.
 - 7: **else**
 - 8: Partition the input into $(1 - \epsilon)$ -alternating batches described in Section 2.4.
 - 9: Run algorithm from Lemma 9 on the $(1 - \epsilon)$ -alternating batches.
 - 10: **end if**
-

$$\min \beta - \alpha$$

$$\forall j \in [1, n] : \sum_{i=1}^n z_{ij} = 1, \quad \forall i \in [1, n] : \sum_{j=1}^n z_{ij} = 1, \quad \forall i, j \in [1, n] : z_{ij} \in \{0, 1\},$$

$$\forall k \in \{1, \dots, n\} : \sum_{j=1}^k \sum_{i=1}^n z_{ij} \cdot x_i - \sum_{j=1}^{k-1} y_j \leq \beta, \tag{8}$$

$$\forall k \in \{1, \dots, n\} : \sum_{j=1}^k \sum_{i=1}^n z_{ij} \cdot x_i - \sum_{j=1}^k y_j \geq \alpha. \tag{9}$$

Observe that (8) and (9) imply that for every interval $I = [k, \ell]$ the sum of the x_i 's assigned to I by Z and the sum of the y_i 's in I differ by at most $\beta - \alpha$. If we replace $z_{ij} \in \{0, 1\}$ with the constraint $z_{ij} \in [0, 1]$, then the solution to the linear program, Z , is an $n \times n$ doubly stochastic matrix. Now we have the following rounding problem. We are given an $n \times n$ doubly stochastic matrix $Z = \{z_{ij}\}$ and we define z_j to be the total fractional value of the x_i 's that are in position j , i.e., $z_j = \sum_{i=1}^n z_{ij} \cdot x_i$. Our goal is to find a permutation of the x_i 's such that the x_i assigned to position j is roughly equal to z_j .

A natural approach would be to decompose Z into a convex combination of permutation matrices and see if one of these gives a good permutation of the elements in X . However, consider the following example:

$$X = \left\{ \underbrace{1, 1, \dots, 1}_{n-k \text{ entries}}, \underbrace{B, B, \dots, B}_k \right\}, \quad \forall i \in [1, n] : y_i = \gamma = \frac{k \cdot B + n - k}{n}.$$

In this case, $z_j = \gamma$ for all $j \in [1, n]$. Thus, a possible decomposition into permutation matrices could look like:

$$\begin{aligned} &\{B, B, \dots, B, 1, 1, \dots, 1, 1\} \\ &\{1, B, B, \dots, B, 1, 1, \dots, 1\} \\ &\quad \dots \\ &\{1, 1, \dots, 1, 1, B, B, \dots, B\}. \end{aligned}$$

Each of these permutations has an interval with very large value, while the optimal permutation of the elements in X is

$$\{1, 1, \dots, 1, B, 1, \dots, 1, B, 1, \dots, 1\}.$$

Algorithm 2 SHIFT($Z, j, i_1, i_2, i_3, \delta$)

```

1:  $\forall i \in \{1, \dots, n\} \setminus \{i_1, i_2, i_3\} : a_i = z_{ij}$ ;
2:  $a_{i_2} = z_{i_2 j} + \delta$ ;
3: if  $x_{i_1} = x_{i_3}$  then
4:    $a_{i_1} = z_{i_1 j} - \delta$ ;    $a_{i_3} = z_{i_3 j}$ ;
5: else
6:    $a_{i_1} = z_{i_1 j} - \delta \cdot \frac{x_{i_2} - x_{i_3}}{x_{i_1} - x_{i_3}}$ ;    $a_{i_3} = z_{i_3 j} - \delta \cdot \frac{x_{i_1} - x_{i_2}}{x_{i_1} - x_{i_3}}$ ;
7: end if
8: return  $a$ 

```

Algorithm 3 TRANSFORM(Z, j, i_1, i_2, i_3)

```

1: The  $j^{\text{th}}$  column of  $Z'$  equals SHIFT( $Z, j, i_1, i_2, i_3, \delta$ ) for  $\delta > 0$  to be chosen later.
2: Let  $j' > j$  denote the smallest index larger than  $j$  with  $z_{i_2 j'} > 0$ . Such an index must exist because row  $i_2$  is not finished in  $Z$  in column  $j$ . The  $(j')^{\text{th}}$  column of  $Z'$  equals SHIFT( $Z, j', i_1, i_2, i_3, -\delta$ ).
3: All columns of  $Z$  and  $Z'$ , except for columns  $j$  and  $j'$ , coincide.
4: The value  $\delta$  is chosen as the largest value for which all entries of  $Z'$  are in  $[0, 1]$ . This value must be strictly larger than 0 due to our choice of  $j, j', i_1, i_2$ , and  $i_3$ .
5: return  $Z'$ 

```

3.1 Transformation

Given a doubly stochastic matrix $Z = \{z_{ij}\}$, we transform it into a doubly stochastic matrix $T = \{t_{ij}\}$ with special properties. First of all, for each j , $z_j = \sum_{i=1}^n t_{ij} \cdot x_i$. This means that if (Z, α, β) is a feasible solution to the linear program then (T, α, β) is also a feasible solution. In particular, if Z is an optimal solution, for which $\beta - \alpha$ is as small as possible, then T is also optimal.

We call a row i in a doubly stochastic matrix $A = \{a_{ij}\}$ *finished* at column ℓ if $\sum_{j=1}^{\ell} a_{ij} = 1$. We say that a matrix T has the *consecutiveness property* if the following holds: for each column j and any rows i_1 and i_3 with $i_1 < i_3$, $t_{i_1 j} > 0$, and $t_{i_3 j} > 0$, each row $i_2 \in \{i_1 + 1, \dots, i_3 - 1\}$ is finished at column j .

Our procedure to transform the matrix Z into a matrix T with the desired property relies on the following transformation rule. Assume that there exist indices j, i_1, i_3 , and $i_2 \in \{i_1 + 1, \dots, i_3 - 1\}$ such that $z_{i_1 j} > 0$, $z_{i_3 j} > 0$, and row i_2 is not finished in matrix Z at column j . Then the procedure SHIFT shown as Algorithm 2 computes a column vector $a = (a_1, \dots, a_n)$, which satisfies the following lemma.

► **Lemma 14.** *For any $\delta \geq 0$, the vector a returned by SHIFT($Z, j, i_1, i_2, i_3, \delta$) satisfies $\sum_{i=1}^n a_i \cdot x_i = z_j$.*

Let Z' denote the matrix that we obtain from Z if we replace the j^{th} column by the vector a returned by the procedure SHIFT. The previous lemma shows that Z' satisfies (8) and (9) for the same β and α as Z because the value z_j is not changed by the procedure. However, the matrix Z' is not doubly stochastic because the rows i_1, i_2 , and i_3 do not add up to one anymore. In order to repair this, we have to apply the SHIFT operation again to another column with $-\delta$. Formally, let us redefine the matrix $Z' = \{z'_{ij}\}$ as the outcome of the operation TRANSFORM shown as Algorithm 3.

Observe that Z' is a doubly stochastic matrix because the rows i_1, i_2 , and i_3 sum up to one and all entries are from $[0, 1]$. Applying Lemma 14 twice implies that (Z', β, α) is a feasible solution to the linear program if (Z, β, α) is one.

71:12 The Alternating Stock Size Problem and the Gasoline Puzzle

We will transform Z by a finite number of applications of the operation TRANSFORM. As long as the current matrix T (which is initially chosen as Z) does not have the consecutiveness property, let j be the smallest index for which there exist indices i_1 , i_3 , and $i_2 \in \{i_1 + 1, \dots, i_3 - 1\}$ such that $t_{i_1 j} > 0$, $t_{i_3 j} > 0$, and row i_2 is not finished in T at column j . Furthermore, let i_1 and i_3 be the smallest and largest index with $t_{i_1 j} > 0$ and $t_{i_3 j} > 0$, respectively, and let i_2 be the smallest index from $\{i_1 + 1, \dots, i_3 - 1\}$ for which row i_2 is not finished at column j . We apply the operation $\text{TRANSFORM}(T, j, i_1, i_2, i_3)$ to obtain a new matrix T .

► **Lemma 15.** *After at most a polynomial number of TRANSFORM operations, no further such operation can be applied. Then T is a doubly stochastic matrix with the consecutiveness property.*

In the remainder, we will not need the matrix Z anymore but only matrix T . For convenience, we will use the notation $t_j = \sum_{i=1}^n t_{ij} \cdot x_i$ instead of z_j even though the transformation ensures that t_j and z_j coincide.

We now define a graph whose connected components or *blocks* will correspond to the row indices from columns that overlap. More formally, let $V = \{1, \dots, n\}$ denote a set of vertices and let G_0 be the empty graph on V . Each column j of T defines a set E_j of edges as follows: the set E_j is a clique on the vertices $i \in V$ with $t_{ij} > 0$, i.e., E_j contains an edge between two vertices i and i' if and only if $t_{ij} > 0$ and $t_{i'j} > 0$. We denote by G_j the graph on V with edge set $E_1 \cup \dots \cup E_j$.

► **Definition 16.** A *block* in G_j is a set of indices in $[1, n]$ that forms a connected component in G_j . A block in G_j is called *finished* if all rows in T corresponding to the indices it contains are finished at column j . Similarly, if a block in G_j contains at least one unfinished row at column j , it is called an *unfinished block*.

If $B \subseteq \{1, \dots, n\}$ is a block in G_j with $i \in B$ then we will say that *block B contains row i* . For the following lemma it is convenient to define a matrix $C = \{c_{ij}\}$, which is the cumulative version of T . To be more precise, the j^{th} column of C equals the sum of the first j columns of T .

► **Lemma 17.** *The following three properties are satisfied for every j .*

1. *Let B be a block in G_j and let $k = \sum_{i \in B} c_{ij}$ denote the value of block B at column j . The number of rows in B is k if B is finished and it is $k + 1$ if B is an unfinished block.*
2. *The set of blocks in G_j emerges from the set of blocks in G_{j-1} by either merging exactly two unfinished blocks or by making one unfinished block finished.*
3. *Let B_1, \dots, B_ℓ denote the unfinished blocks in G_j . Then there exist non-overlapping intervals $I_1, \dots, I_\ell \subseteq [1, n]$ with $B_i \subseteq I_i$ for every i .*

One might ask if the consecutiveness property is satisfied by every optimal extreme point of the linear program. Let us mention that this is not the case. A simple counterexample is provided by the instance $X = \{9, 6, 4, 1\}$ and $Y = \{5, 5, 5, 5\}$. In this instance, an optimal extreme point would be, for example, to take one half of each of the items x_1 and x_4 in steps one and three and to take one half of each of the items x_2 and x_3 in steps two and four. This extreme point does however not satisfy the consecutiveness property. Hence, the transformation described in this section is necessary.

3.2 Rounding

In this section, we use the transformed matrix T to create the solution matrix R , which is a doubly stochastic 0/1 matrix, i.e., a permutation matrix. We apply the following rounding method.

- 1: **for** $j = 1$ to n **do**
- 2: Let B denote the *active* block in G_j , i.e., the block that contains the rows i with $t_{ij} > 0$.
- 3: Let p denote the smallest index in B such that $r_{pi} = 0$ for all $i < j$.
- 4: Set $r_{pj} = 1$ and $r_{qj} = 0$ for all $q \neq p$.
- 5: **end for**

Observe that the first step is well-defined because all non-zero entries in column j belong by definition to the same block of G_j . The resulting matrix R will be doubly stochastic, since each column contains a single one, as does each row. We just need to prove that in Line 3 there always exists a row $p \in B$ that is unfinished in R at column $j - 1$. This follows from the first part of the next lemma because, due to Lemma 17, the active block B in G_j emerges from one or two unfinished blocks in G_{j-1} and these blocks each contain a row that is unfinished in R at column $j - 1$.

► **Lemma 18.** *Let B be a block in G_j for some $j \in \{1, \dots, n\}$.*

1. *If B is an unfinished block in G_j and p is the largest index in B , then $r_{pi} = 0$ for all $i \leq j$ and all rows corresponding to $B \setminus \{p\}$ are finished in R at column j .*
2. *If B is a finished block in G_j , then for all $q \in B$, row q is finished in R at column j .*

We define the *value* of a permutation matrix M to be the smallest γ for which there exist α' and β' with $\gamma = \beta' - \alpha'$ such that (M, α', β') is a feasible solution to the linear program.

► **Theorem 19.** *Let (T, α, β) be an optimal solution to the linear program. Then $(R, \alpha, \beta + \mu_x)$ is a feasible solution to the linear program. Hence, the value of the matrix R is at most $(\beta - \alpha) + \mu_x \leq 2 \cdot OPT$, where OPT denotes the value of the optimal permutation matrix.*

For ease of notation, we define r_j as follows: $r_j = \sum_{i=1}^n r_{ij} \cdot x_i$. Note that r_j corresponds to the value of the element from Y that the algorithm places in position j . We will see later that Theorem 19 follows easily from the next lemma.

► **Lemma 20.** *For each $k \in \{1, \dots, n\}$,*

$$\sum_{j=1}^k (r_j - t_j) \in [0, \mu_x]. \quad (10)$$

We need the following lemma in the proof of Lemma 20.

► **Lemma 21.** *Let b be the largest index in an unfinished block B in G_j . Then,*

$$c_{bj} = \sum_{i \in B \setminus \{b\}} (1 - c_{ij}).$$

Proof. Let the value of the unfinished block B be $k = \sum_{i \in B} c_{ij}$. By property 1 of Lemma 17, block B consists of $k + 1$ rows. Thus, we have:

$$c_{bj} = k - \sum_{i \in B \setminus \{b\}} c_{ij} = \sum_{i \in B \setminus \{b\}} (1 - c_{ij}). \quad \blacktriangleleft$$

71:14 The Alternating Stock Size Problem and the Gasoline Puzzle

Proof of Lemma 20. Let us consider the sets of finished and unfinished blocks in G_k , \mathcal{B}_F and \mathcal{B}_U , respectively. For a block $B \in \mathcal{B}_F \cup \mathcal{B}_U$, we denote by

$$\text{er}_k(B) = \sum_{i \in B} \sum_{j=1}^k x_i(r_{ij} - t_{ij})$$

its rounding error. Since each row is contained in exactly one block of G_k ,

$$\sum_{j=1}^k (r_j - t_j) = \sum_{j=1}^k \sum_{i=1}^n x_i(r_{ij} - t_{ij}) = \sum_{i=1}^n \sum_{j=1}^k x_i(r_{ij} - t_{ij}) = \sum_{B \in \mathcal{B}_F \cup \mathcal{B}_U} \text{er}_k(B). \quad (11)$$

Hence, in order to prove the lemma, it suffices to bound the rounding errors of the blocks.

If block B is finished in G_k , then all rows that belong to B are finished in T and in R (due to property 2 of Lemma 18) at column k . Hence,

$$\text{er}_k(B) = \sum_{i \in B} \sum_{j=1}^k x_i(r_{ij} - t_{ij}) = \sum_{i \in B} x_i \cdot \left(\sum_{j=1}^k r_{ij} - \sum_{j=1}^k t_{ij} \right) = \sum_{i \in B} x_i \cdot (1 - 1) = 0. \quad (12)$$

Now consider an unfinished block B in G_k , and let a and b denote the smallest and largest index in B , respectively. By Lemma 18, all rows in the block except for b are finished in R at column k (i.e., $\sum_{j=1}^k r_{ij} = 1$ for $i \in B \setminus \{b\}$ and $\sum_{j=1}^k r_{bj} = 0$). The rounding error of B can thus be bounded as follows (remember that $c_{ik} = \sum_{j=1}^k t_{ij}$):

$$\begin{aligned} \text{er}_k(B) &= \sum_{i \in B} \sum_{j=1}^k x_i(r_{ij} - t_{ij}) = \sum_{i \in B} x_i \sum_{j=1}^k r_{ij} - \sum_{i \in B} x_i \sum_{j=1}^k t_{ij} \\ &= \sum_{i \in B \setminus \{b\}} x_i - \sum_{i \in B} x_i c_{ik} = \sum_{i \in B \setminus \{b\}} x_i(1 - c_{ik}) - x_b c_{bk} \\ &= \sum_{i \in B \setminus \{b\}} x_i(1 - c_{ik}) - x_b \sum_{i \in B \setminus \{b\}} (1 - c_{ik}) \end{aligned} \quad (13)$$

$$\begin{aligned} &= \sum_{i \in B \setminus \{b\}} (x_i - x_b)(1 - c_{ik}) \\ &\leq (x_a - x_b) \sum_{i \in B \setminus \{b\}} (1 - c_{ik}) \end{aligned} \quad (14)$$

$$= (x_a - x_b) \cdot c_{bj} \quad (15)$$

$$\leq x_a - x_b. \quad (16)$$

Equations (13) and (15) follow from Lemma 21. Inequality (16) follows from the fact that $c_{bj} \leq 1$. Inequality (14) follows from the facts that $1 - c_{ik} \geq 0$ and $x_i - x_b \geq 0$ for all $i \in B$. These facts also imply that $\text{er}_k(B) \geq 0$. Hence,

$$\text{er}_k(B) \in [0, x_a - x_b]. \quad (17)$$

Together (11) and (12) imply

$$\sum_{j=1}^k (r_j - t_j) = \sum_{B \in \mathcal{B}_F \cup \mathcal{B}_U} \text{er}_k(B) = \sum_{B \in \mathcal{B}_F} \text{er}_k(B) + \sum_{B \in \mathcal{B}_U} \text{er}_k(B) = \sum_{B \in \mathcal{B}_U} \text{er}_k(B). \quad (18)$$

Now, let B_1, \dots, B_h denote the unfinished blocks in G_k , and for each block B_f in \mathcal{B}_U , let a_f and b_f denote the minimum and maximum indices, respectively, contained in the block.

Property 3 of Lemma 17 implies that the intervals $[a_f, b_f]$ are pairwise disjoint. Hence, (17) implies

$$\sum_{B \in \mathcal{B}_U} \text{er}_k(B) \in \left[0, \sum_{f=1}^h (x_{a_f} - x_{b_f}) \right] \subseteq [0, x_1 - x_n] \subseteq [0, \mu_x].$$

Together with (18) this implies the lemma. ◀

Now we are ready to prove Theorem 19.

Proof of Theorem 19. Let (T, α, β) denote an optimal solution to the linear program. By definition, our rounding method produces a permutation matrix R . Lemma 20 implies that $(R, \alpha, \beta + \mu_y)$ is also a feasible solution to the linear program because for each $k \in \{1, \dots, n\}$,

$$\sum_{j=1}^k x_j - \sum_{j=1}^{k-1} \sum_{i=1}^n r_{ij} \cdot x_i = \sum_{j=1}^k x_j - \sum_{j=1}^{k-1} r_j \leq \sum_{j=1}^k x_j - \sum_{j=1}^{k-1} t_j + \mu_x \leq \beta + \mu_x$$

and

$$\sum_{j=1}^k x_j - \sum_{j=1}^k \sum_{i=1}^n r_{ij} \cdot x_i = \sum_{j=1}^k x_j - \sum_{j=1}^k r_j \geq \sum_{j=1}^k x_j - \sum_{j=1}^k t_j \geq \alpha.$$

Now the theorem follows because $\text{OPT} \geq \mu_x$ and $\text{OPT} \geq \beta - \alpha$. ◀

4 Conclusions

We have introduced two new variants of the stock size problem and have presented non-trivial approximation algorithms for them. The most intriguing question for our variants as well as for the original stock size problem is if the approximation guarantees can be improved. Each of these problems is NP-hard but no APX-hardness is known. So it is conceivable that there exists a PTAS. Closing this gap seems very challenging.

We note that the additive integrality gap of the linear program in Section 3 can be arbitrarily close to μ_y . Consider the following instance:

$$x = \frac{(n-1) + \mu}{n}, \quad X = \underbrace{\{x, \dots, x\}}_{n \text{ entries}}, \quad Y = \underbrace{\{\mu, 1, 1, \dots, 1\}}_{n-1 \text{ entries}}.$$

Then the value of the linear program is x . However, the optimal value is μ , which can be arbitrarily larger than x .

Acknowledgments. We would like to thank Anupam Gupta for several useful examples and enlightening conversations. We would like to thank Jochen Könemann for pointing out a connection between the alternating stock size problem and the optimization version of the gasoline puzzle. We would also like to thank Tamás Kis for pointing out related papers on scheduling problems.

Part of this work was done during the trimester program on Combinatorial Optimization at the Hausdorff Institute for Mathematics in Bonn and we would like to thank HIM for their organization and hospitality during the program.

References

- 1 H. M. Abdel-Wahab and T. Kameda. Scheduling to minimize maximum cumulative cost subject to series-parallel precedence constraints. *Operations Research*, 26(1):141–158, 1978.
- 2 Wojciech Banaszczyk. The steinitz constant of the plane. *Journal für die Reine und Angewandte Mathematik*, 373:218–220, 1987.
- 3 Imre Bárány. On the power of linear dependencies. In *Building bridges*, pages 31–45. Springer, 2008.
- 4 Jacek Blazewicz, Jan Karel Lenstra, and AHG Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.
- 5 Dirk Briskorn, Byung-Cheon Choi, Kangbok Lee, Joseph Leung, and Michael Pinedo. Complexity of single machine scheduling subject to nonnegative inventory constraints. *European Journal of Operational Research*, 207(2):605–619, 2010.
- 6 J Carlier and AHG Rinnooy Kan. Scheduling subject to nonrenewable-resource constraints. *Operations Research Letters*, 1(2):52–55, 1982.
- 7 V. S. Grinberg and S. V. Sevastyanov. The value of the Steinitz constant. *Functional Analysis and its Applications*, 14:56–57, 1980.
- 8 Péter Györgyi and Tamás Kis. Approximation schemes for single machine scheduling with non-renewable resource constraints. *Journal of Scheduling*, 17(2):135–144, 2014.
- 9 Péter Györgyi and Tamás Kis. Approximability of scheduling problems with resource consuming jobs. *Annals of Operations Research*, 235(1):319–336, 2015.
- 10 Nick Harvey and Samira Samadi. Near-optimal herding. In *COLT*, pages 1165–1182, 2014.
- 11 Hans Kellerer, Vladimir Kotov, Franz Rendl, and Gerhard J. Woeginger. The stock size problem. *Operations Research*, 46(3):S1–S12, 1998.
- 12 L. Lovász. *Combinatorial Problems and Exercises*. North-Holland, 1979.
- 13 Clyde L. Monma. Sequencing to minimize the maximum job cost. *Operations Research*, 28(4):942–951, 1980.
- 14 Ehab Morsy and Erwin Pesch. Approximation algorithms for inventory constrained scheduling on a single machine. *Journal of Scheduling*, 18(6):645–653, 2015.
- 15 Klaus Neumann and Christoph Schwindt. Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, 56(3):513–533, 2003.
- 16 Alantha Newman, Heiko Röglin, and Johanna Seif. The alternating stock size problem and the gasoline puzzle. *arXiv:1511.09259*, 2015.