Master Operations Research and Combinatorial Optimization
Master Mathématiques & Applications

Iterative Rounding algorithms for the Extended Gasoline Problem

19/02/2024 - 26/07/2024

Lucas Lorieau

tutored by

Moritz Mühlenthaler
Grenoble INP - Ensimag, UGA

Heiko Röglin
University of Bonn

Defended on the 26/06/2024 before a jury composed of

Prof Nadia Brauner
Prof Van-Dat Cung
Prof Heiko Röglin
Prof Moritz Mühlenthaler
Prof Alantha Newman

Acknowledgement

First and foremost, I would like to deeply thank my supervisor Heiko Röglin for the constant support he was able to bring me before and during my stay in Bonn, and for the inspiring work sessions we had throughout the internship. Working with him was a real pleasure and I'm glad that I could undergo my first academic experience (and first abroad professional experience too!) under his supervision.

Special thanks to Moritz Mühlenthaler for being my supervisior back in France and helping me while I was looking for a position for my master's thesis. Without him, I might not have found such an interesting topic, especially abroad. I also thank him for his reassuring words during some of our meetings.

I would also like to thank some of the professors of the ORCO master in Grenoble for their valuable teachings and discussions that were a true motivation for my studies. In particular, I want to thank Nadia Brauner, Van-Dat Cung and Alantha Newman, that will join the defense jury of this thesis. Many thanks to Zoltán Szigeti for his everlasting cheerfulness during his talks and for introducing me to the world of combinatorics and graph theory during my studies at Ensimag.

I would like to thank my colleagues at the Institute of Computer Science in Bonn for the multiple exchanges we have had together and for having me feel a part of the institute since the very beginning of my stay. Thanks to Sarah, Joshua, Jan, Rebecca, Alex, Anurag and all the other ones.

Thanks to my friends Logan, Antoine, Adèle, Maxime and Olympe for supporting me from a distance during this long stay in an unknown country. A special acknowledgement to Jolan for helping me training for my defense and for his kind and empowering words. Finally, I owe very special thanks to my family and especially my parents, that have been an endless source of strength and motivation since the very beginning of my studies.

Abstract

The Gasoline Problem is related to scheduling with non-renewable resources constraints. In this thesis, we study an approximation algorithm called "Iterative Rounding" that was introduced by Rajković (2022) for two variants of the problem: the first one in one dimension for which a 2-approximation is presented by Newman et al. (2018), and the other one in 2 or more dimensions that was also introduced by Rajković (2022). A conjecture in this article claims that the algorithm is a 2-approximation for both variants. We show that the approximation factor of Iterative Rounding algorithm cannot be lower than 2 by describing a type of instances for which the approximation factor can be made arbitrarily close to 2. We also present guarantees on the approximation factor of a Greedy algorithm on a special case. Experimental results that led to analyse the behavior of the Iterative Rounding algorithm are presented for both variants of the problem, and followups for the analysis of the 2 dimensions and more version of the problem are proposed.

Résumé

Le *Gazoline Problem* est un problème d'ordonnancement avec contraintes de ressources non renouvelables. Dans ce rapport, nous étudions un algorithme d'approximation dit « *Iterarive Rounding* » décrit par Rajković (2022) pour deux variantes de ce problème ; l'une en une dimension pour laquelle une 2-approximation est présentée dans Newman et al. (2018) et l'autre en 2 dimensions et plus, également introduite par Rajković (2022). Il est conjecturé dans cet article que l'algorithme est une 2-approximation pour chacune des variantes du problème. Nous montrons que l'algorithme *Iterative Rounding* ne peut avoir un facteur d'approximation plus petit que 2 en décrivant une famille d'instances pour laquelle le facteur d'approximation peut être rendu arbitrairement proche de 2. Nous prouvons également dans un cas particulier des garanties sur le facteur d'approximation d'un algorithme Glouton. Des résultats expériementaux ayant permis d'analyser le comportement de l'algorithme *Iterative Rounding* sont présentés pour les deux versions du problème, et des pistes d'analyses pour la version en 2 dimensions et plus sont proposées.

Contents

— 1 —
Introduction

Among one of the most widely represented domains of Combinatorial Optimization and Operational Research, Scheduling enables to finely tune industrial processes. The intensive research in this domain led to the development of powerful tools that are pivotal in managing complex industrial systems, especially when subtle constraints have to be addressed to obtain feasible solutions.

The Scheduling term specifically deals with problems where tasks have to be assigned to agents (*e.g.* machines), with the goal to find an optimal assignment given a specific criterion. This assignment task is sometimes critical for the industrial process to run smoothly and is sometimes the bottleneck preventing improvements in efficiency. A significant part of scheduling problems are NP-complete, which means only approximation algorithms are known for polynomial time solving of such problems.

We will focus throughout this thesis on the "scheduling with non renewable resources" framework. In this type of problems, a stock of "rare" resource is one of the main constraints of the problem. This stock is used for the consuming tasks we have to schedule and has a finite value, and it has to be restocked by using specific orders that also have to be scheduled accordingly. Of course, consuming tasks cannot be scheduled if the stock is not sufficient. This type of problem has been studied first by Carlier et al. (1982) in the early 1980's, and continues to be studied because of the numerous applications of this framework.

One of those applications is the Stock size problem studied by Kellerer et al. (1998). In this problem, we can imagine that a factory using a specific rare material has some orders to fulfill, each of those consuming a specific amount of the material. It also has some replenishment orders available, to refill its stock of product. All orders mentioned before have a fixed value of needed (respectively delivered) material. Of course, having a stock is costly and the factory wants to have at any point of time the smallest stock possible to limit its expenses. The goal is then to schedule the different orders so that the stock is always sufficient to process the incoming consuming task, and the maximum stock over the time period is minimized. One could also want to schedule a delivery each day, and thus enforcing the constraint of having *alternating* delivery orders and replenishment orders. Newman et al. (2018) introduce this variation of the problem and prove the first results on this new problem.

One can also consider to only be allowed to permute one of the two sequence of values, then creating another problem, called the Gasoline Problem. This problem was first introduced following a puzzle presented by Lovász (1979) and will be the one we focus on throughout this thesis. This problem shares common structure with different classical problems, such as bin packing, bin covering, or discrepancy problems. In particular, one can think about this problem as a "dual version" of the prefix discrepancy problem (or signed series problem). Some of these related problems will be briefly stated in Section 1.3.

The first part of this thesis will define properly the problem, and give some insights on problems related to the Gasoline Problem as well as presenting some known results concerning this problem and its generalisations. The next section will define one generic approximation algorithm solving the Gasoline Problem, called iterative rounding, and will present its differences with usual iterative rounding procedures. It is worth to study this algorithm because it can be quite easily adapted when dealing with generalisations of the Gasoline problem unlike other known algorithms tackling this kind of problems. We will first present some numerical results that helped getting some intuition on the behavior of the algorithm, and we will focus specifically on trying to understand on what type of instance our algorithm can struggle to compute a good approximation of the optimal solution. Then, we will give some results on the 1D version of the Gasoline problem. Among them, we prove that the approximation factor of the Iterative Rounding is at least 2. We also consider a subcase of the problem, for which we were able to give some guarantee on the approximation factor for a Greedy algorithm. Finally, we also state some remarks on the extended version of the problem, and present again some numerical results that might also help to get more insight on the behavior of the algorithm for this extended version of the problem.

1.1 Problem statement

The first appearance of a puzzle from which originates the Gasoline problem is due to Lovász (1979) and is presented as follows:

> 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.

Kellerer et al. (1998) naturally formulate a general problem from this puzzle. Suppose that each road segment is associated with a positive integer representing the amount of fuel necessary to travel on the said segment, and that each station can refill a fixed positive integer amount of gasoline. Let us also assume that there is exactly one spot for a station between each road segment, and that every spot has to be occupied by one station. We are allowed to choose the final spot of all stations, and our goal is to minimise the size of the gasoline tank of our car so that we can travel through the speed track without running out of fuel. This problem called the Stock Size problem, was the first studied (see Section 1.3) after the Lovász statement cited above.

The problem considerd in this thesis is slightly different and we can use another simple example to illustrate it. Consider a factory that produces every day goods that are send in the evening. For one good produced, the factory has to consume one ingredient. Each morning, the factory has the possibility to order a certain amount of ingredients. If the factory orders too much, it can store the overload in its stock, and can use the stored ingredients afterwards. Typically, the stock can be used if the factory does not order a sufficient amount of ingredients to fulfill the demand of produced goods on a day. Finally, over a specified period of time, the factory knows for each day how many goods their customers require, and also knows the different quantities of ingredients it can order from its suppliers. The goal is then to decide each day which ingredient order to ask for in order to minimize the span of the stock over the period. This objective is rather easy to understand : if the span is small, then the factory do not have to buy or rent a big storage space and will spare money.

Let us define more formally the problem described above. We are given as input two sequences of positive integers $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ with $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$. The order of y_i values being fixed, our goal is to find a permutation π that minimises the maximum prefix sum discrepancy $\eta = \beta - \alpha$, where

$$\sum_{i=1}^{k} x_{\pi(i)} - \sum_{i=1}^{k-1} y_i \leq \beta \qquad \text{for } 1 \leq k \leq n; \tag{1}$$

$$\sum_{i=1}^{k} x_{\pi(i)} - \sum_{i=1}^{k} y_i \geq \alpha \qquad \text{for } 1 \leq k \leq n. \tag{2}$$

For a fixed value of k, we will call the left-hand side expression of (1) a *major prefix sum* (and we denote it by S_k) and the left-hand-side expression of (2) a *minor prefix sum* (denoted by s_k). We will also denote by μ_x (respectively μ_y) the maximum value of the X (respectively Y) sequence, and define $\mu = \max(\mu_x, \mu_y)$. Observe that μ is a lower bound on the value of η. We model this problem with a Mixed Integer Linear Program, whose definition is presented in Figure 1.1.

This original version of the problem is based on two lists of scalars. One can also imagine some extension of the problem where vectors in l dimensions are considered, and a distance function over $\mathbb{R}^k$ is used for the discrepancy objective while keeping other characteristics of the problem unchanged. We will call this problem the Extended Gasoline Problem. Of course, defining a MILP for this problem might not be possible depending on the distance function we will use for the objective, as this objective might not be expressed as a linear expression. We will mainly consider the case of l_1 in this thesis, as it can be easily described in terms of linear expressions and have practical interpretation. Indeed, if one thinks back to the example of Lovász, we could consider a vehicule consuming several types of fuel at the same time, in different quantities on each part of the track. The goal would be then to minimise the total volume of fuel tanks, which would correspond to using the l_1 norm as the objective of the problem.

Let us introduce notations for this extended problem. Now, we specifically denote by $x_i^{(p)}$ (respectively $y_i^{(p)}$) the p-th coordinate of the i-th value of the sequence X (respectively Y). Other notations are also adapted when necessary, with the addition of $\cdot^{(p)}$ to indicate the coordinate of the value considered. Furthermore, we will denote the norm used in the objective by $\|\cdot\|$. The associated MILP is displayed in Figure 1.2. It can be seen as a repetition of the program of Figure 1.1 on each coordinate, with a general objective (*i.e.* an objective encompassing all coordinates).

1.2 Previous results

The main results about the Gasoline problem can be found in Newman et al. (2018). They prove that the problem is NP-complete with a reduction from the 3-partition problem. They also present a 2-approximation algorithm operating in two steps, first computing a specific doubly stochastic matrix presenting a "consecutiveness" property, and then using a rounding technique to transform the previous matrix into a permutation matrix. Guarantees arise from *ad-hoc* arguments based on the specific structure associated with consecutiveness. This algorithm relies heavily on sorting the values of the X sequence and cannot be adapted to the extended version of the problem in a reasonable fashion.

The only previous mention of the generalised problem can be found in Rajković (2022). The generalised problem is defined, as well as an Iterative Rounding algorithm for both 1D and generalized versions of the problem. This algorithm will be studied later in Chapter 2. No major results are present in the thesis concerning the generalized problem, but 2 conjectures that we will examine are stated.

$$\min \beta - \alpha \qquad \text{s.t.} \tag{3}$$

$$\sum_{j=1}^{n}\sum_{i=1}^{k} x_i z_{ij} - \sum_{i=1}^{k-1} y_i \le \beta \qquad \text{for } 1 \le k \le n \tag{4}$$

$$\sum_{j=1}^{n}\sum_{i=1}^{k} x_i z_{ij} - \sum_{i=1}^{k} y_i \ge \alpha \qquad \text{for } 1 \le k \le n \tag{5}$$

$$Z\mathbb{1} \le \mathbb{1} \tag{6}$$

$$\mathbb{1}^T Z \le \mathbb{1}^T \tag{7}$$

$$z_{ij} \in \{0,1\} \qquad \text{for } 1 \le i,j \le n \tag{8}$$

$$\alpha, \beta \in \mathbb{R} \tag{9}$$

Figure 1.1: MILP formulation for the Gasoline problem

$$\begin{aligned}
&\min \|\beta-\alpha\|_1 \qquad \text{s.t.} && \\
&\sum_{j=1}^{n}\sum_{i=1}^{k} x_i^{(p)} z_{ij} - \sum_{i=1}^{k-1} y_i^{(p)} \le \beta^{(p)} && \text{for } 1 \le k \le n, \forall p \le l \\
&\sum_{j=1}^{n}\sum_{i=1}^{k} x_i^{(p)} z_{ij} - \sum_{i=1}^{k} y_i^{(p)} \ge \alpha^{(p)} && \text{for } 1 \le k \le n, \forall p \le l \\
&Z\mathbb{1} \le \mathbb{1} && \\
&\mathbb{1}^T Z \le \mathbb{1}^T && \\
&z_{ij} \in \{0,1\} && \text{for } 1 \le i,j \le n \\
&\alpha,\beta \in \mathbb{R}^l &&
\end{aligned}$$

Figure 1.2: MILP formulation for the Generalised Gasoline Problem

Conjecture 1.2.1 (Rajković 2022, Conjecture 1)**.** *The Iterative Rounding algorithm has an approximation ratio of 2.*

Conjecture 1.2.2 (Rajković 2022, Conjecture 3)**.** *The Multidimensional Iterative Rounding algorithm has an approximation ratio of 2.*

Those conjectures are supported by numerical experiments that were performed in order to describe the average behavior of the algorithm on randomly constructed instances of the gasoline problem in 1D and 2D cases. The study of the previous conjectures is the main motivation of this thesis, especially because no approximation algorithm is known for the extended version of the Gasoline problem. Moreover, little to no research has been conducted on the “bad instances” of the Gasoline problem, and finding constructions of instances for which an interesting lower bound on the approximation ratio of the Iterative Rounding algorithm is reached would be the first step to understand better which kind of instance is difficult to optimise on.

1.3 Related work

The Gasoline problem is closely related to the combinatorial discrepancy problems. In those kind of problems, a ground set of vectors $E \in \mathbb{R}^n$ is given, and we consider a set of subsets of E, denoted by $\mathcal{S}$. The goal is to assign each vector $v \in E$ a sign $c(v) \in \{-1,1\}$ (*i.e.* find a two coloring of E) so that the quantity

$$\operatorname{disc}(c) = \max_{S\in\mathcal{S}} \left\| \sum_{v\in S} c(v)v \right\|$$

is minimized, with $\|\cdot\|$ being a norm over $\mathbb{R}^d$. The first formal definition of this class of problems is found in Matoušek (1999), but some problems contained in this class were studied well before. Main works of this field are focusing on results concerning the maximal discrepancy of the system considered, rather than on computing the actual discrepancy of a given system. The proofs associated are for the majority non constructive, like with the “partial coloring” technique, a widespread technique in the field of discrepancy theory. One example of a major result in discrepancy is the Beck-Fiala theorem that can be found in Beck et al. (1981). This paper deals with the case where each element of the ground set is at most present t times throughout the sets of $\mathcal{S}$. Under this constraint, the discrepancy of unitary vectors can be bounded by $2t-1$. Another well known result is stated in Spencer (1985) and gives a bound of discrepancy on systems where the number n of sets considered equals the number of elements of the ground set. In this case, the bound is in $O\left(\sqrt{n}\right)$. The discrepancy theory is a rich domain with extended research on some sub-problems, and has applications in either very specific topics like numerical integration or random trial formulation but also in more widespread subjects like theoretical results about the Divide-and-Conquer algorithm.

When we consider $\mathcal{S}_\pi = \{S_i \mid S_i = \{v_{\pi(1)}, \ldots, v_{\pi(i)}\}, 1 \le i \le |E|\}$ for some permutation π, we then obtain a "dual" problem of the Gasoline problem: instead of trying to find a permutation minimizing the discrepancy of prefix sums while the sign of each vector is fixed, the goal is to find the optimal sign assignment for a specific permutation. This problem has been named the Prefix Discrepancy problem and was introduced by Spencer (1977). Banaszczyk (2012) gives a $O\left(\sqrt{\log(d)+\log(n)}\right)$ bound for the maximal discrepancy of unit vectors, where d is the dimension of the vectors and n the length of the sequence of vectors. The problem is also generalised in Bansal et al. (2022) to prefixes in a directed acyclic graph and some negative results are also given.

Some work concerning the related problems of the Stock Size problem and the Alternating Stock size problem can also be found in the literature. As stated before, Kellerer et al. (1998) studied the Stock Size problem and proved it to be NP-hard. They describe some approximation algorithms to solve the problem, ranging from a simple sequencing-based 2 approximation to a more involved algorithm resorting to a dual bin packing approach resulting in a $\frac{3}{2}$ approximation factor. The alternating version of the problem is first defined in Newman et al. (2018). They prove that it is also NP-hard and give 2 approximation algorithms for this problem, one involving a pairing procedure yielding an approximation factor of 2, and present a more involved algorithm whose approximation factor is 1.79.

— 2 —
Iterative Rounding algorithm

2.1 Definition and remarks

The Iterative Rounding algorithm that we will study in this section is related, in a way, to classical Iterative rounding procedures. The algorithms of this type are based on linear relaxation of integer linear programs. Such a relaxation is computed at each iteration, and either a rounding on variables that are close to be integer is performed, or some constraints that are only slightly violated are relaxed. Some argument depending on the studied problem and classical linear programming results ensures that those operations can be applied at each iteration. This framework can be applied to a wide range of different combinatorial problems, and can be used to provide efficient and elegant approximation algorithms for a lot of NP-hard problems. For a deeper analysis of this technique see Lau et al. (2011).

In our case, the general process of the algorithm is somewhat different. We will heavily rely on the linear relaxation of our MILP defined in Figure 1.1, like for classical Iterative Rounding algorithms. However, the considered algorithm solve multiple linear programs fixing one value $z_{i,j}$ at a time to 1, and keeps for this iteration the added constraint that yielded the best value for its iteration.

The intuitive goal of this algorithm lies in a balance between preventing an immediate degradation in the value of the computed solution and preventing a *future degradation*. To do so, at some step i the relaxed LP is used to detect whether a possible choice at step i will provoke a big deterioration in the value of the solution in the future steps. For instance, it could be possible that incoming Y values sum to a large value, and it might be better to choose at step i a smaller X value so that afterwards, the biggest X values are available to prevent the prefix sums from becoming too low and increasing the value of the approximated solution. In that sense, Algorithm 1 is an improved version of the basic greedy algorithm: it will as well prevent choosing a trivially bad value (*i.e.* a one that will increase directly the value of the partial solution because either S_i or s_i is becoming the new maximal (respectively minimal) prefix sum), but will also prevent choosing a value that will not worsen directly the solution, but would provoke a big increase afterwards, whatever the future choice would be.

Algorithm 1 : Iterative Rounding

```
    $I \leftarrow \{1, \dots, n\}$
    LP $\leftarrow LP^*$
    **for** $j = 1$ to $n$ **do**
        optValue = $+\infty$
5:      optIndex = $-1$
        **for** $i \in I$ **do**
            NewLP $\leftarrow$ LP + "$z_{i,j} = 1$"
            value $\leftarrow$ optimal value of NewLP
            **if** value < optValue **then**
10:             optValue $\leftarrow$ value
                optIndex $\leftarrow$ $i$city
            **end if**
        **end for**
        LP $\leftarrow$ LP + "$z_{\text{optIndex},j} = 1$"
15:     $I \leftarrow I \backslash \{\text{optIndex}\}$
    **end for**
```

2.2 Comparison to the Greedy algorithm

The structure of the Iterative Rounding algorithm seems very close to a Greedy algorithm (some pseudo-code describing this algorithm is presented in Algorithm 2). Indeed, at each step, the Iterative Rounding algorithm chooses a value for the next unfilled slot of the solution, based on the previous choices he made and the remaining values. In that regard, it shares with the Greedy algorithm the property that it tries to avoid immediate losses in the value of the computed solution. Indeed, this is not sufficient to obtain an algorithm with a bounded approximation ratio. Consider the next instance

$$X = \{\underbrace{2, \dots, 2}_{\frac{n}{4}\text{ times}}, \underbrace{1, \dots, 1}_{\frac{n}{2}\text{ times}}, \underbrace{0, \dots, 0}_{\frac{n}{4}\text{ times}}\},$$

$$Y = \{\underbrace{2, 0, \dots, 2, 0}_{\frac{n}{2}\text{ times}}, \underbrace{2, \dots, 2}_{\frac{n}{4}\text{ times}}, \underbrace{0, \dots, 0}_{\frac{n}{4}\text{ times}}\}.$$

By placing first all 1 values, then all 2 values and finally all 0 values, one can verify that this instance has an optimal value of 2. But the greedy algorithm will place all 2 and 0 values (by alternating them) first, before placing the remaining 1 values. It is also easy to see that this solution will create a value of $\frac{n}{4}$, thus producing an approximation factor of $\frac{n}{2} \xrightarrow[n \to +\infty]{} +\infty$.

The increase in complexity of the Iterative Rounding procedure is then necessary: future consequences are to be assessed if the algorithm is to produce good solutions, even on pathological instances. This is the role of the LP relaxation used in the Iterative rounding, as explained in the previous section.

Algorithm 2 : Greedy algorithm

```
    $s \leftarrow 0$
    for $i$ in $1, \dots, n$ do
        Choose an available value $v$ of $X$ so that $|s + v - y_i|$ is minimised fo slot $i$
        Make $v$ unavailable
5:  end for
```

2.3 Numerical experiments

To get a first idea of the behavior of the algorithm and further understanding what type of instance can be hard to solve, some numerical experiments were conducted. Those experiments consist in solving both exactly and with the approximation algorithm some randomly-generated instances and compute the ratio between the optimal and approximated values. There are two different goals for this type of experiment. First, we want to compute some experimental lower bound on the approximation ratio, to give us insight on the best approximation ratio possible for the algorithm. On top of that, finding such lower bound means that we also get the instance achieving this ratio and both optimal and approximated solution for this specific instance. With this, observations on several results can lead us to conjecture some results before proving it. It appears that this technique lead to find the lower bound proved in Proposition 3.1.1.

2.3.1 General behavior

As a first step, some of the experiments of Rajković (2022) were reproduced. In particular, we compute mean and maximal approximation ratio of some randomly generated instances in order to check that the results of Rajković (2022) and to confirm some intuition on the behavior of the algorithm. To do so, we generate instances by using a simple procedure. The size of the sequence X and Y is fixed, and the sequences are initialized with zeroes. Then, we will repeat some simple random modification k times in order to generate the final instance. This modification consist in drawing uniformly one value of each of the sequence, and choose with probability $\frac{1}{2}$ to either increment both values or decrement them. By doing so, we end up with two sequences of integers that each sums to the same value. If one value in either sequence is negative, we begin again the process until we end up with some valid instance.

Those results are displayed in Table 2.1a. We used the same conditions as in the original thesis, namely using Python3 as our coding language and using Gurobi as a MILP solver. Furthermore, for each size of the instance n (*i.e.* the size of the X sequence) we used $N = 10000$ instances. The results we obtain are similar to those presented in Rajković (2022), and confirms that it is difficult to obtain instances with an approximation ratio above around 1.7 using local search and small instances. In particular, we also observe that the approximation ratio does not significantly increase with the size of the instance when the values are also bounded. This might indicate

n	Max	Mean	σ	% of non-optimal
5	1.5	1.036	0.006	6.77
10	1.667	1.085	0.011	18.30
15	1.667	1.100	0.012	22.06
20	1.667	1.111	0.013	24.89

(a) Slot Ordered algorithm

n	Max	Mean	σ	% of non-optimal
5	1.5	1.033	0.005	5.81
10	1.667	1.085	0.011	19.36
15	1.667	1.116	0.015	28.32
20	1.667	1.135	0.016	33.06

(b) Value Ordered algorithm

Table 2.1: Results of random generation of instances experiments

that varying one parameter at a time light not be sufficient. As a small remark, one can think about modifying the Iterative Rounding algorithm so that instead of choosing iteratively which value will be placed in a specific slot (we call this algorithm "Slot Ordered"), the algorithm (called "Value Ordered") chooses first where the first value will be placed, then the second value, etc. As one can see in Table 2.1b, there is no real difference with the original Iterative Rounding algorithm, which lead us not to consider only the Slot ordered version for the rest of this thesis.

2.3.2 Local Search

In order to reach the best lower bound possible, we considered a second type of experiment. Some local search has been used to improve iteratively our results. Local searches are mainly used in optimisation problems when the search space has some structure. Indeed, this type of searches relies on the idea that the value of solutions has some continuous property, and thus that making step by step local improvement leads in the end to reaching a (local) optimum. This type of search is easy to implement and can be adapted to a lot of different contexts, including our problem. The algorithm used for our local search is displayed in Algorithm 3 and consists in the following: at each step a current best instance is maintained, and modified slightly with some random noise. We obtain then a new instance that we can solve both optimally and with the Iterative Rounding algorithm. We compute then its approximation ratio, and if it is greater than the current best, the current best instance is updated by replacing it with the modified instance. The noise used in line 7 to modify the instance is produced with

Algorithm 3 : Local search of a bad instance

```
    Generate a random instance $I$
    Compute the approximation ratio $\rho$ hit by instance $I$
    currentInstance $\leftarrow I$
    currentRatio $\leftarrow \rho$
5:  $i \leftarrow 1$
    while $i < N$ do
        Modify randomly $I$ to obtain a new instance $I'$
        Compute the approximation ratio $\rho'$ hit by $I'$
        if $\rho' > \rho$ then
10:         currentInstance $\leftarrow I'$
            currentRatio $\leftarrow \rho'$
        end if
        $i \leftarrow i+1$
    end while
15: return currentInstance
```

the exact same process that was used previously to generate new instances randomly. Of course, whenever we generate and apply a noise that creates a negative value in one of the two sequence of the modified instance, we discard the change and generate new noises until the modified instance is valid.

Before presenting and commenting the results of the experiments, let us highlight that the behavior of the algorithm can vary greatly on two neighboring instances, meaning that the local search might not be as efficient as for other classical problems it is used for. Indeed, let us consider the next instance (X,Y) defined as follows

$$X = [12, 10, 1, 1, 12, 19, 18, 30, 22, 26, 30, 29, 30, 29, 0];$$
$$Y = [21, 25, 15, 17, 8, 10, 6, 3, 11, 21, 26, 26, 29, 27, 24].$$

Consider also a slightly modified instance (X',Y') defined as follows (modified values are in bold)

$$X' = [12, 10, 1, 1, 12, 19, \mathbf{17}, 30, 22, 26, 30, 29, 30, 29, 0];$$
$$Y' = [21, 25, 15, 17, 8, 10, 6, 3, 11, \mathbf{20}, 26, 26, 29, 27, 24].$$

As one can remark, those two instances varies only from a small amount: only one value in each sequence is decremented by one. Intuitively, one might think that the approximation algorithm should have the same behavior for the two instances, or at least that the differences should not be too important, since the instances are very similar to each other. Figure 2.1 shows on the contrary that the behavior of the Iterative Rounding is visibly different between the two instances. For each instance, the red plot correspond to the optimal solution, and the blue one to the solution produced by the Iterative Rounding algorithm. If the optimal solution seems almost identical for both instances, the approximated solution is on the contrary quite different: for the (X,Y)

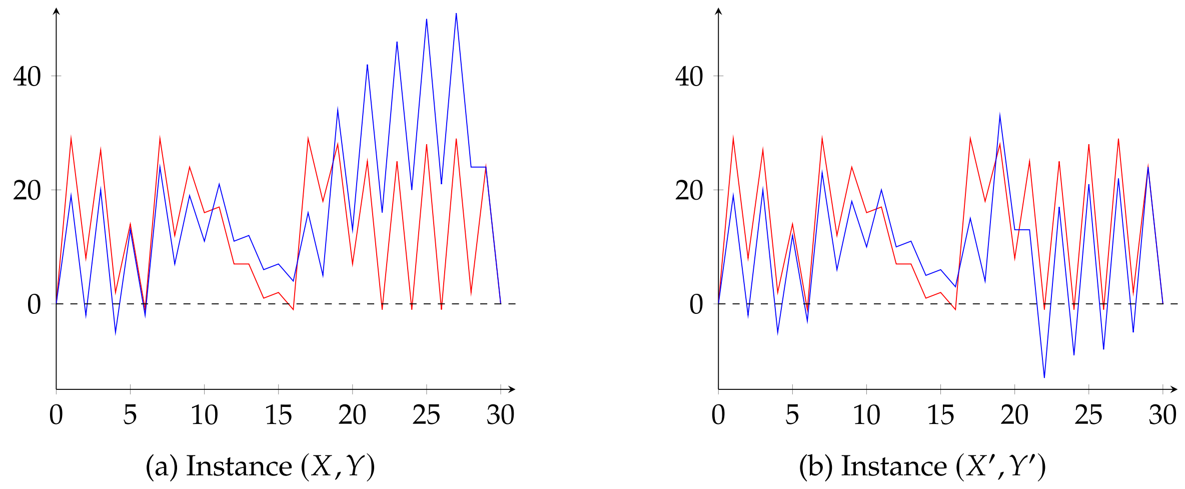


Figure 2.1: Comparison of the behavior of Algorithm 1 on two close instances

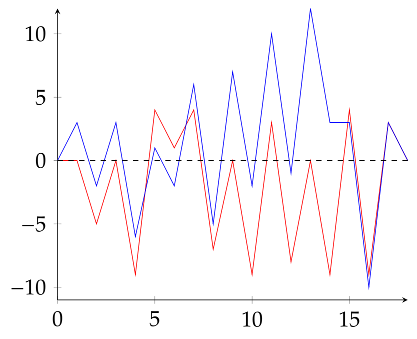
instance, the value of the solution is 56, but for the (X', Y') instance the value is 46. This lead do a decrease of the approximation ratio from 1.86 to 1.57.

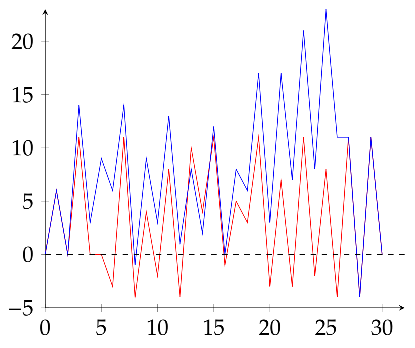
This example shows that the "neighborhood" of one instance can present quite large variation of approximation ratio. From this observation, one can then understand that the Local search can become inefficient as we reach higher approximation ratio: some neighbors of the current best instance will have a smaller approximation ratio, slowing down the whole search process and preventing it from reaching a local optimum value close to the global one in a reasonable time. Nevertheless, our Local Search algorithm managed to compute better lower bounds than the best one given in Rajković (2022), and those are presented in Figure 2.2.

Each instance found thanks to the Local Search algorithm shares some common properties, that seems to be creating the large gap between the optimal solution and the approximated solution. First, each instance contains in the X sequence a 0 value. If we observe the optimal solution of each of those instances, the 0 value tends to be placed in the very first slots of the solution. On the contrary, the 0 value is rather located in the last slots of the approximated value. In general, the Y sequence contains also substantially higher values in the end of the sequence, compared with the other Y values. This has a consequence on the general outline of the approximated solutions: one can notice that, before the 0 is reached in the permutation computed by the Iterative Rounding algorithm, the value of the different S_i prefix sums gets quite high, with a peak just before reaching the 0 value. This peak is seemingly what is causing the high approximation ratio hit by those type of instance. Those observations reveal some intuition on the type of instance for which the Iterative Rounding algorithm is not performing well, and lead ultimately to the proof of Proposition 3.1.1 in the next section.

Example	Sequence	Values
Small Instance	X	[3, 5, 7, 0, 8, 12, 13, 12, 13]
	Y	[5, 9, 3, 11, 9, 11, 9, 13, 3]
Medium Instance	X	[6, 14, 6, 0, 8, 10, 15, 7, 10, 8, 10, 14, 14, 11, 15]
	Y	[6, 11, 3, 15, 6, 12, 6, 12, 2, 14, 10, 13, 12, 15, 11]
Big Instance	X	[13, 14, 7, 0, 7, 9, 17, 8, 5, 3, 6, 17, 14, 11, 10, 23, 22, 22, 22, 21, 21]
	Y	[6, 17, 3, 13, 7, 13, 11, 12, 4, 11, 12, 10, 13, 14, 9, 18, 18, 20, 20, 22, 19]

Table 2.2: Instances obtained by using Algorithm 3


(a) Small instance, ratio = 1.69


(b) Medium instance, ratio = 1.80

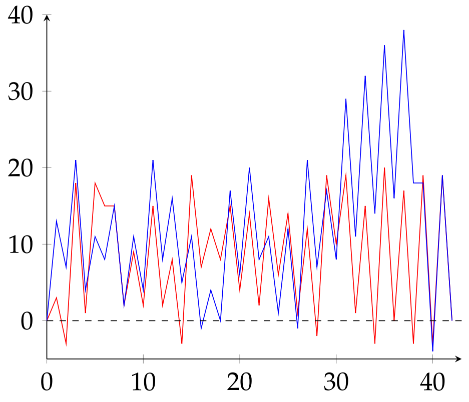

(c) Big instance, ratio = 1.83

Figure 2.2: Graphical representation of instances obtained by using Algorithm 3

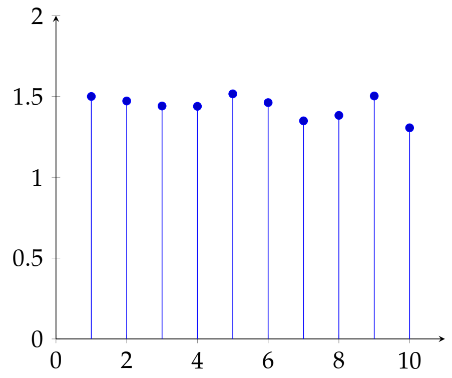


Figure 2.3: Values of the highest approximation ratio found by Local Search for the several number of dimensions

2.3.3 Generalised problem

Finally, some experiments were conducted in order to assess the behavior of the algorithm on the generalised problem, especially for high dimensions. The goal was to support Conjecture 1.2.2 by trying to find the worst possible instances for a given dimension of the problem and see whether those bad instances could yield an approximation ratio greater than 2. We adapted the Local Search algorithm presented previously and used it to generate bad instances. In order to keep the execution time reasonable, we performed 5000 iterations of the Local search on randomly generated instances of size 10 (*i.e.* with 10 entries in each sequence X and Y) for each dimension from 2 to 10. Results are displayed in Figure 2.3. One can notice that the maximum approximation ratios obtained for the several dimensions tested are quite close one to another, but seem to be decreasing for increasing dimension. This tends to confirm the conjecture previously cited, as finding bad instances randomly seems not to be easier when several dimensions are considered instead of only one. Yet, one could also argue that augmenting the dimension of the problem is also augmenting drastically the combinatorial complexity, and so it might be necessary to also augment the number of iteration of the Local Search in order to obtain similar results.

One must still mitigate the previous claim concerning the behavior of the algorithm. Indeed, the new structure induced by considering several dimensions might enable new behaviors that we could not directly spot during the Local Search procedure. It requires some more investigations that go beyond the scope of this thesis, but this is a natural followup to the work presented here. In particular, determine whether the 2D case is behaving in a similar way as the 1D case in terms of approximation could already be very instructive on this question.

— 3 —

Approximation results

This chapter will present all the original results of the thesis. The main result concerning the lower bound on the approximation ratio of the Iterative Rounding algorithm for the 1D case will be presented first. Then, the presentation of an upper bound result for a restricted case or the problem will be given. The chapter ends with a quick commentary on results for the Generalised version of the problem.

3.1 One dimension version

3.1.1 Lower bound

In this section, we introduce a construction of instances for the Gasoline Problem for which the approximation ratio of the Iterative Rounding algorithm is arbitrarily close to 2, proving the next proposition.

Proposition 3.1.1. *The Iterative Rounding algorithm has an approximation ratio greater or equal than 2.*

To construct such an instance, let us define a specific type of instances for which the behavior of Iterative Rounding is easily described.

Definition 3.1.1. *Let n be an non-negative integer. An instance (X,Y) of the Gasoline Problem is called an n-*staircase instance *if there exists a finite sequence of non-negative integers $(a_i)_{0\leq i\leq k}$ so that*

$$X = (a_0,\dots,a_k,\underbrace{n,\dots,n}_{n-1\text{ times}},0),$$
$$Y = (a_0,\dots,a_k,\underbrace{n-1,\dots,n-1}_{n\text{ times}}),$$
$$\forall\, 0\leq i\leq k,\ a_i\leq n-1.$$

This type of instance is defined with two different blocks. In the first one, the same values are present in X and in Y while in the second, that we will call the *staircase block*, values and their number of occurrences are chosen so that the only value present in the Y sequence is $n-1$, a single 0 is present in the X sequence, and X and Y still sum to the same value. This construction is motivated by the observations of the previous section. Indeed, one 0 value is present in the X sequence, and the biggest values of the Y sequence are placed at the very end of it. Observe that because of the third property of n-staircase instances, $\mu = n$ is a lower bound on the optimal value of those instances. The next proposition gives some insight on the behavior of the Iterative Rounding algorithm on those instances.

Proposition 3.1.2. *Let n be a non-negative integer. The Iterative Rounding algorithm always yields a solution of value at least $2(n-1)$ for any n-staircase instance of the Gasoline Problem.*

Recall that the Iterative Rounding algorithm uses the relaxed version of the MILP defined in Figure 1.1 whose feasible solutions are doubly stochastic matrices. Furthermore, we also recall that $\mu_y = \max(Y)$ is a lower bound of the optimal solution of the linear relaxation of the MILP. To prove Proposition 3.1.2, we provide an optimal solution to the relaxed MILP, and argue that this prevents the algorithm from placing the 0 value of the n-staircase instance before the staircase block.

Proof. Let n be a non-negative integer, and (X, Y) a n-staircase instance, *i.e.*

$$X = (a_0, \dots, a_k, \underbrace{n, \dots, n}_{n-1 \text{ times}}, 0) \text{ and } Y = (a_0, \dots, a_k, \underbrace{n-1, \dots, n-1}_{n \text{ times}}).$$

Let us consider the following block matrix

$$M = \begin{pmatrix} I_{k+1} & 0 \\ 0 & \frac{1}{n}\mathbb{I}_n \end{pmatrix} \quad \text{where } \mathbb{I}_n = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} \text{ is an } n \times n \text{ matrix.}$$

This matrix is clearly doubly stochastic, so it is a solution of the relaxed MILP formulation of the Gasoline problem. Let us show that this solution has a value of $n-1 = \mu_y$, and thus is optimal for this relaxed MILP. First, if $0 \le l \le k$, we have that $S_l = \sum_{j=0}^{l} a_j - \sum_{j=0}^{l-1} a_j = a_l \le n-1$. If $k < l \le k+n-1$, then $S_l = \sum_{j=k+1}^{l} \sum_{i=k+1}^{k+n-2} \frac{1}{n} \times n - \sum_{j=k}^{l-1} n - 1 = n-1$. Furthermore, we have for all l that $s_l = 0$, and this instance is then of value $n-1$.

By recurrence, let us prove that the algorithm sets the value a_i at the i-th slot for $0 \le i \le k-1$. For $i = 0$, the first constraint considered is "$z_{0,0} = 1$". When it is added to our MILP, M remains a feasible solution, and thus the value associated with a_0 for this iteration is minimum. Because the algorithm picks the first X-value yielding the minimum solution-value, a_0 will be chosen at this iteration. Let us suppose next that the first i values have been already chosen during the i first iterations of the algorithm. On the $(i+1)$-th iteration, the first unchosen value is then a_{i+1}. When we consider this value at current iteration, the MILP is appended with constraints "$z_{j,j} = 1$" for

$0 \le j \le i+1$. M is still a feasible solution for this MILP, so for the same reasons the algorithm will choose the value a_{i+1} at this iteration.

Observe that the previous statement implies that the value 0 is never placed during the first k iterations. This means that it will be placed between two $n-1$ values of the Y sequence. This ensures that the value of the solution obtained is of value at least $2(n-1)$. □

The previous proposition describes the behavior of the Iterative Rounding algorithm, but gives no insight on the optimal value of the n-staircase instances. Consider the next example

$$X = (n, n, \ldots, n, 0), \;\; Y = (n-1, \cdots, n-1)$$

for n a non negative integer. Indeed, this instance is n-staircase with an empty first block. Its optimal value is clearly $2(n-1)$ and the Iterative Rounding algorithm would yield an optimal solution for this instance. We thus have to consider n-staircase instances with low optimal value. We define a new tool that will be used to describe such an instance.

Definition 3.1.2. *Given a fixed sequence of positive integers $Y = (y_0, \ldots, y_l)$ and n a non-negative integer, we say that a sequence of positive integers $X = (x_0, \ldots, x_l)$ forms an n-*valid pattern *if*

$$i)\; \sum_{i=0}^{l} x_i = \sum_{i=0}^{l} y_i;$$

$$ii)\; \sum_{i=0}^{k} x_i - \sum_{i=0}^{k-1} y_i \le n \text{ for } 0 \le k \le l;$$

$$iii)\; \sum_{i=0}^{k} x_i - \sum_{i=0}^{k} y_i \ge 0 \text{ for } 0 \le k \le l-1.$$

Valid patterns, similarly to batches in Kellerer et al. (1998), can be seen as components of "good" solutions to an instance: consider $Y_0, \ldots, Y_{j-1}$ sequences of numbers, and let us denote Y the concatenation of those sequences. If X can be partitioned (without keeping order) into j sequences $X_0, \ldots, X_{j-1}$ so that each (X_i, Y_i) is forming an n-valid pattern, then the optimal solution of the instance (X, Y) has a value at most n.

We will next construct an n-staircase instance which can be decomposed into n-valid patterns. This will ensure that the instance has an optimal value sufficiently low to prove Proposition 3.1.1. This instance is composed of regular patterns whose components are linked by a simple recurring formula.

Proposition 3.1.3. *Let k be a non-negative integer and $(u_i)_{0\leq i\leq k}$ be a non-constant sequence of positive integers. The following statements are equivalent*

i) The patterns $P_i = \big((2^k, u_i), (u_{i+1}, u_{i+1})\big)$ for $0 \leq i \leq k$ are 2^k-valid;

ii) $u_i = 2^k(1-2^{-i})$ for all $0 \leq i \leq k$.

Proof. Suppose i). Then, for all $0 \leq i \leq k$, $2^k + u_i = 2u_{i+1}$. We recognise an arithmetico-geometric sequence of the form $u_{i+1} = au_i + b$ with $a = \frac{1}{2}$ and $b = 2^{k-1}$. The closed form formula for u_i is then

$$\begin{aligned} u_i &= a^i(u_0 - \frac{b}{1-a}) + \frac{b}{1-a} \\ &= 2^{-i}(u_0 - 2^k) + 2^k. \end{aligned}$$

The first major prefix sum of a 2^k-valid pattern for P_0 then gives the constraint $u_0 \leq 2^k$, and because all u_i must be integers, we have in particular for $i = k$ that $u_0 - 2^k = 2^k p$ with p an integer. We deduce from those two observations that $u_0 = 0$ or $u_0 = 2^k$. Because the sequence (u_i) is not constant, $u_0 = 0$ and the general term of the sequence is $u_i = 2^k(1-2^{-i})$.

Now, suppose ii). For $0 \leq i \leq k$, we have $u_i = 2^k - 2^{k-i}$, so all u_i are positive integer. Furthermore, (u_i) is clearly non-constant, and by construction of the sequence in the first part of the proof, $2^k + u_i = 2u_{i+1}$. Finally, let us consider a pattern P_i for some $0 \leq i \leq k$. We have

$$\begin{aligned} 2^k - u_{i+1} &\geq 0 && \text{because } (u_i) \text{ is increasing and } u_k = 2^k - 1; \\ 2^k - u_{i+1} + u_i &\leq 2^k && \text{because } (u_i) \text{ is increasing.} \end{aligned}$$

This means that all prefix conditions are respected, and that all P_i are 2^k-valid. □

An example of the previously defined patterns is represented in Figure 3.1. The patterns P_i are exactly the components that we will use to build the instance in the proof of Proposition 3.1.1.

Proof of Proposition 3.1.1. Let k be a non-negative integer and $u_i = 2^k(1-2^{-i})$ for all $0 \leq i \leq k$. Consider the following instance

$$X = (\underbrace{u_1, u_1}_{2^1\text{ times}}, \underbrace{u_2, \dots, u_2}_{2^2\text{ times}}, \dots, \underbrace{u_{k-1}, \dots, u_{k-1}}_{2^{k-1}\text{ times}}, \underbrace{2^k, \dots, 2^k}_{2^k-1\text{ times}}, u_0);$$

$$Y = (\underbrace{u_1, u_1}_{2^1\text{ times}}, \underbrace{u_2, \dots, u_2}_{2^2\text{ times}}, \dots, \underbrace{u_{k-1}, \dots, u_{k-1}}_{2^{k-1}\text{ times}}, \underbrace{u_k, \dots, u_k}_{2^k\text{ times}}).$$

Recall that $u_0 = 0$, $u_k = 2^k - 1$, and $u_i \geq 0$ so by definition (X, Y) is a 2^k-staircase instance. By Proposition 3.1.2, the Iterative Rounding algorithm outputs a solution of value at least $2(2^k - 1)$ when run on the instance (X, Y).

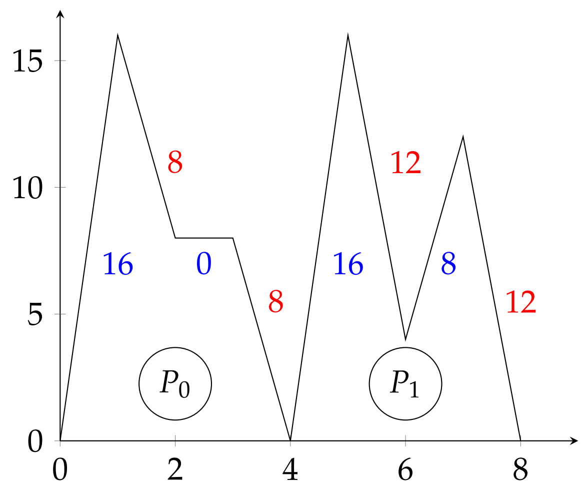


Figure 3.1: Patterns P_0 and P_1 for $k = 4$

Consider now the following partition (considering order) $Y = (Y_1, \dots, Y_k)$ with $Y_i = \underbrace{(u_i, \dots, u_i)}_{2^i \text{ times}}$. Each Y_i can then also be decomposed in $2^i - 1$ couples $Y_{i,j} = (u_i, u_i)$ for $1 \le j \le 2^{i-1}$. Let us define a partition of X (without considering order) in the following way

$$X = \bigcup_{\substack{1 \le i \le k \\ 1 \le j \le 2^{i-1}}} X_{i,j} = (2^k, u_i - 1).$$

First, let us argue that the $X_{i,j}$ form actually a partition of X. For each $1 \le i \le k$ the 2^{i-1} sets $X_{i,j}$ are identical, formed of one 2^k value and one u_{i-1} value. Each u_i is then present 2^i times exactly and each 2^k is present $\sum_{i=1}^k 2^{i-1} = 2^k - 1$ times exactly in this union, so the $X_{i,j}$ are effectively a partition of X.

Then, observe the patterns $(X_{i,j}, Y_{i,j})$ for $1 \le j \le 2^{i-1}$ are equal to the P_i pattern of Proposition 3.1.3. By the same proposition, because of the definition of (u_i), the patterns P_i are 2^k-valid. This means that we found a decomposition into 2^k-valid patterns of the instance (X, Y), and its optimal value is at most 2^k. Because $\mu = 2^k$ for this instance, the optimal value of (X, Y) is exactly 2^k. The approximation ratio of the Iterative Rounding algorithm on this instance is then $r_k \ge \frac{2(2^k-1)}{2^k} \xrightarrow[k \to +\infty]{} 2$. □

As a side note, for some fixed k the instance (X, Y) is following the behaviors we identified in Section 2.3.2. Indeed, there is always one 0 value in X, and it is always placed at position 2 for the optimal solution and at the last position for the approximated one. Also, the biggest values of the Y sequence are located at the end of the sequence, and they represent almost half of the sequence. One complete graphical representation of the optimal and approximated solutions for the value $k = 3$ is given in Figure 3.2.

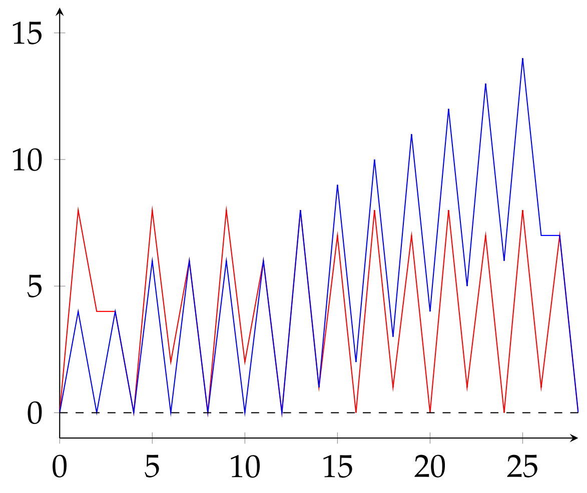


Figure 3.2: Optimal and approximated solutions for the instance (X, Y) and $k = 3$

Proposition 3.1.1 implies that The Iterative Rounding algorithm will not improve on the algorithm described in Newman et al. (2018). Yet, it is still interesting to study it because of two main reasons. First, the algorithm is significantly simpler than the current 2-approximation algorithm known. Indeed, the procedure used in Newman et al. (2018) is quite involved, and the rounding technique described is somewhat not intuitive to implement, because of its complexity. Furthermore, this algorithm cannot be extended easily to the extended version of the Gasoline problem. As explained in Section 1.2, the known approximation algorithm relies heavily on sorting the X values, and this could be difficult to adapt for the case were we consider vectors instead of plain scalars. On the contrary, Algorithm 1 can be easily adapted to deal with vectors instead of scalars.

3.1.2 Upper bound for the $\{1, K\}$ subcase

To study the upper bound of the approximation ratio of our algorithm, let us consider a simpler subproblem. In the latter, the X sequence has values in $\{1, K\}$ with $K > 1$. This means that the permutation we want to find is equivalent to choosing where to place the 1 values. We also slightly restrict the values of the Y sequence so that its values are all positive. We call this restricted problem the $\{1, K\}$-Gasoline Problem. Interestingly, we prove for this subproblem that the greedy algorithm achieves a 2 approximation ratio.

Proposition 3.1.4. *Algorithm 4 is a 2-approximation for the $\{1, K\}$-Gasoline Problem.*

The Iterative Rounding can be considered as an improved Greedy algorithm, we thus conjecture that the same type of result apply for the Iterative Rounding (Note that the next conjecture is of course included in the strongest Conjecture 1.2.1).

Algorithm 4 : Greedy algorithm for the $\{1,K\}$-Gasoline Problem

```
    $s \leftarrow 0$
    $i \leftarrow 0$
    while there is both 1 and $K$ values available in X do
        if $s \leq 0$ then
5:          Choose value $K$ for $x_i$
            $s \leftarrow s + K - y_i$
        else
            Choose value 1 for $x_i$
            $s \leftarrow s + 1 - y_i$
10:     end if
        $i \leftarrow i + 1$
    end while
    for $k \in \{i, \dots, n\}$ do
        Choose remaining value for $x_k$
15: end for
```

Conjecture 3.1.1. *Algorithm 1 is a 2-approximation for the $\{1,K\}$-Gasoline Problem.*

To prove the previous proposition, we will use a more elaborated lower bound on the optimum. We will denote it v, and we define it as follows

$$v = \max_{\substack{1 \leq a \leq b \leq n \\ b-a \leq \#K}} \left(\sum_{i=a}^{b} y_i - (b-a)K \right).$$

Lemma 3.1.1. *The value v is a lower bound on the optimal value of any instance of the $\{1,K\}$-Gasoline Problem.*

Proof. Consider the two indices a and b for which the maximum is obtained in the definition of v, and an optimal permutation π^*. By definition of the optimum value, we have

$$\begin{aligned} \text{OPT} \geq S_a - s_b &= \sum_{i=a}^{b} y_i - \sum_{i=a+1}^{b} x_{\pi^*(i)} \\ &\geq \sum_{i=a}^{b} y_i - \sum_{i=a+1}^{b} K = v. \end{aligned}$$ □

This lower bound encompasses the μ lower bound we previously considered. Indeed, if y_k is the maximum value of the Y sequence we obtain $\mu = y_k = \sum_{i=k}^{k} y_i - (k-k)K \leq v$. It is more precise because it can handle cases where some part of the Y sequence is significantly more "dense" than the X sequence can be, inducing a mandatory increase in the optimal value of the instance.

Consider j the minimal index for which the sequence output by the greedy algorithm is constant. The step j of the Greedy algorithm is thus the moment of the last "real" decision made by the algorithm, because only one type of value in X is still available in further steps. This remark defines two different phases in the execution of the algorithm: first decisions can be made, and after the step j no decisions are really done. We will bound the values α and β for both phases, ensuring that the Greedy algorithm has a approximation factor of 2.

Proof of Proposition 3.1.4. Let us consider an instance (X,Y) of the $\{1,K\}$-Gasoline Problem, and denote by $(x'_i)_{1\le i\le n}$ the X sequence after applying the permutation computed by the Greedy algorithm on (X,Y). Let be j the minimal index for which $(x'_i)_{j\le i\le n}$ is constant. Consider also a non-negative index $i \le j$ contained in the first phase where both 1 and K values are available to choose.

By the definition of the algorithm, if $s_{i-1} \le 0$, the algorithm will choose to assign a K value for x'_i. This means that $S_i = s_{i-1} + x'_i \le K \le \mu$. Consider now the maximal index $p < i$ so that s_p is non-negative (exists because $s_0 = 0$). We have then $s_p - s_i = \sum_{k=p}^{i} y_k - x'_k \le \sum_{k=p}^{i} y_k - (i-p+1)K \le \nu$. This means, because s_p is supposed non negative, that $s_i \ge -\nu$.

If $s_{i-1} > 0$, the algorithm will assign the value 1 to x'_i. Then, we have

$$s_i = s_{i-1} + 1 - y_i \le s_{i-1}. \tag{1}$$

Consider the minimal index $p \le i$ so that $s_p > 0$ (remark that $p \ge 1$ because $s_0 = 0$). Then, by definition $s_{p-1} \le 0$ and $x'_p = K$. We thus obtain $s_p = s_{p-1} + K - y_p \le \mu - 1$ and using Equation (1) recursively, $s_k \le \mu - 1$ for all $p \le k \le i$ and in particularly, $s_i \le \mu - 1$ *i.e.* $S_i \le \mu$. Finally, we have $s_i = s_{i-1} + 1 - y_i \ge -y_i \ge -\mu$. This means that for all $i \le j$, we have $S_i \le \nu \le \mathrm{OPT}$ and $s_i \ge -\nu \ge -\mathrm{OPT}$.

Now, let us tackle the phase when only one type of values is still available in X. Suppose that the type of remaining values is 1. Then, we clearly have that $(s_k)_{j\le k\le n}$ is decreasing and $s_k \le \mu - 1$ for all $k \ge j$, *i.e.* $S_k \le \mu$. Furthermore, for all $j \le k \le n$, $s_k \ge s_n = 0$. Consider now the type of remaining values to be K. First, let p be the index so that $s_p = \min_{j\le k\le n}(s_k)$. By considering again a maximal index $k \le p$ with $s_k \ge 0$, we obtain that $s_k \ge -\nu$. Now, let us consider $q \ge j$ the index so that $S_q = \max_{j\le k\le n}(S_k)$ and $k \ge j$ the minimal index so that $s_k \le 0$ (exists because $s_n = 0$). We have then that $S_q - s_k = \sum_{i=q}^{k} y_i - K(k-q) \le \nu$, so $S_q \le \nu$. Finally, for this second phase we have for all $i > j$, $S_i \le \nu \le \mathrm{OPT}$ and $s_i \ge -\nu \ge \mathrm{OPT}$. This means that for all $i \le n$, $S_i \le \mathrm{OPT}$ and $s_i \ge -\mathrm{OPT}$, so we also obtain $\beta_{\mathrm{Greedy}} \le \mathrm{OPT}$ and $\alpha_{\mathrm{Greedy}} \ge -\mathrm{OPT}$. We can thus conclude that $\beta_{\mathrm{Greedy}} - \alpha_{\mathrm{Greedy}} \le 2\mathrm{OPT}$. □

This result is also interesting to understand further from where the complexity of the Gasoline Problem arises. As stated before, the Greedy algorithm can be arbitrarily bad for approximating the unrestricted problem, even if we consider only 3 values instead of 2 for the X sequence. There is thus a real change in the complexity of the problem when restricted to this version.

3.2 Generalised version

This section will quickly describe the results that can be stated for the generalised version of the problem. As the workload of the thesis has been mostly focused towards finding results for the 1D case first, the majority of the results of this section derives from results stated in the previous section.

Concerning the lower bound on the approximation ratio of the Iterative Rounding algorithm, a simple argument can be used to prove the next proposition based on the instances described to prove Proposition 3.1.1.

Proposition 3.2.1. *The Iterative Rounding algorithm for the Generalized Gasoline problem has an approximation ratio greater or equal than 2.*

For a recall, the notation $x_i^{(l)}$ is still used to denote the l-th coordinate of the i-th element of the sequence X, and is adapted to other sequences of numbers.

Proof. Consider a dimension $m \in \mathbb{N}$ for the values of X and Y, and let k be a positive integer. Consider the following instance

$$x_i^{(l)} = \begin{cases} x_i^* \text{ if } l = 1, \\ 0 \text{ otherwise;} \end{cases}$$
$$y_i^{(l)} = \begin{cases} y_i^* \text{ if } l = 1, \\ 0 \text{ otherwise;} \end{cases}$$

with $(x_i^*)_i$ and $(y_i^*)_i$ being the sequences of the instance (X, Y) defined in the proof of Proposition 3.1.1 for the value k. The analysis performed in that proof is also applicable on this one for the first dimension of the x_i and y_i values. Furthermore, because all other values are equal to 0, only the first dimension contribute to the increases in the objective value, thus ensuring that the approximation ratio of this instance is the same as the approximation ratio of the instance (X, Y) for the 1D version of the problem. □

Considering the conclusions of the experiments made for the high dimension versions of the problem that confirms Conjecture 1.2.2, one can imagine that this instance is also sufficient to tighten the approximation factor if the conjecture holds.

No specific result on the upper bound for the Generalised Gasoline problem have been looked for, since no results in the general case for the 1D Gasoline problem were proved. Indeed, the behavior of the objective seems to be quite different both for l_0 and l_1 norms. In the first case, we consider only the max stock value of any dimension and the minimum stock value also for any, which means those extrema can be obtained for two different dimensions of the problem, and thus the analysis of the approximation guarantees cannot be directly applied. This is the same for the l_2 norm, for which we sum the discrepancy on each dimension, which differs even more from the objective used in the 1D version of the problem.

— 4 —
Conclusion

In this thesis, some approximation algorithms have been studied to understand their behavior for several variants of the so called Gasoline problem. After defining those variants, namely the "plain" Gasoline problem, the $\{1, K\}$ subcase and the Generalised Gasoline problem, we described a new heuristic called Iterative Rounding algorithm to tackle those problem. Its definition is simpler than the currently known best approximation algorithm for the Gasoline problem, and the Iterative Rounding algorithm can also be applied to the generalised version of the problem unlike the latter.

For understanding better the behavior of the algorithm and try to gain insight on a possible analysis of its approximation guarantees, experiments were conducted to design bad instances in term of approximation gap yielded by the Iterative Rounding. Confirming previous conjectures on this algorithm claiming that its approximation factor is 2, a Local Search technique produced random instances hitting an approximation factor around 1.9. After observing the instances obtained through those experiments, a construction of a family of instances for the "plain" version has been described, yielding the negative result that the Iterative Rounding could not have a lower approximation factor than 2. On the positive side, some guarantee for the Greedy algorithm was proven for the $\{1, K\}$ subcase. Finally, some results on the generalised version of the problem are deduced from the work on the "plain" version.

Some clear followups to this thesis can be identified. Indeed, the most important one is to prove some guarantee on the Iterative Rounding algorithm. The $\{1, K\}$ subcase seems promising, as a part of the proof presented for the Greedy algorithm might be reused for the Iterative Rounding. Other interesting questions can be formulated after the work presented in the thesis. For instance, it could be interesting to find other subcases of the Gasoline problem that can be approximated with a constant factor, like for the $\{1, K\}$ variation. The same question can also be generalised to the multidimensional version of the problem. One last question deals with the lower bound instances for the generalised Gasoline problem. As stated during the thesis, the change of structure in the definition of the problem might allow to find new instances that could yield approximation factors greater than 2, despite the experiments results that tends to indicate that the Iterative Rounding keeps for the generalised version of the problem the same general behavior and approximation factor than for the "plain" version. Indeed, tweaking the Iterative Rounding to get better approximation could also be imagined. The lower bound instances described in this thesis can give insights on the weaknesses of the algorithm, and thus trying to mitigate those could lead to a better approximation algorithm.

Bibliography

Banaszczyk, Wojciech (May 2012). “On series of signed vectors and their rearrangements”. In: *Random Structures & Algorithms* 40, pp. 301–316.

Bansal, Nikhil et al. (2022). “Prefix Discrepancy, Smoothed Analysis, and Combinatorial Vector Balancing”. In: *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Ed. by Mark Braverman. Vol. 215. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 13:1–13:22. ISBN: 978-3-95977-217-4.

Beck, József and Tibor Fiala (1981). ““Integer-making” theorems”. In: *Discrete Applied Mathematics* 3.1, pp. 1–8. ISSN: 0166-218X.

Carlier, J. and A.H.G. Rinnooy Kan (1982). “Scheduling subject to nonrenewable-resource constraints”. In: *Operations Research Letters* 1.2, pp. 52–55. ISSN: 0167-6377.

Kellerer, Hans et al. (1998). “The Stock Size Problem”. In: *Operations Research* 46.3, S1–S12. ISSN: 0030364X, 15265463.

Lau, Lap Chi, R. Ravi, and Mohit Singh (2011). *Iterative Methods in Combinatorial Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press.

Lovász, László (1979). *Combinatorial Problems and Exercises*. AMS/Chelsea publication. North-Holland Publishing Company. ISBN: 9780821869475.

Matoušek, Jiří (1999). “Combinatorial Discrepancy”. In: *Geometric Discrepancy: An Illustrated Guide*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 101–135. ISBN: 978-3-642-03942-3.

Newman, Alantha, Heiko Röglin, and Johanna Seif (Apr. 2018). “The Alternating Stock Size Problem and the Gasoline Puzzle”. In: *ACM Trans. Algorithms* 14.2. ISSN: 1549-6325.

Rajković, Ivana (Mar. 2022). “Approximation Algorithms for the Stock Size Problem and the Gasoline Problem”. Master’s thesis.

Spencer, Joel (1977). “Balancing games”. In: *Journal of Combinatorial Theory, Series B* 23.1, pp. 68–74. ISSN: 0095-8956.

— (1985). “Six Standard Deviations Suffice”. In: *Transactions of the American Mathematical Society* 289.2, pp. 679–706. ISSN: 00029947.