Thèse de doctorat en informatique
de Sorbonne Université

École Doctorale Informatique, Télécommunications et Électronique

---

An Optimization Perspective on the Construction of Low-Discrepancy Point Sets

---

François Clément

présentée et soutenue publiquement
pour obtenir le grade de
Docteur en Sciences Informatique

Date provisoire de soutenance: 18 juillet 2024

Directeurs de thèse

Dr. Carola Doerr
CNRS, Sorbonne Université, LIP6, France

Dr. Luís Paquete
CISUC, DEI, University of Coimbra, Portugal

Rapporteurs

Pr. Dmitriy Bilyk
University of Minnesota, United States of America

Priv.-Doz. Mag. Dr. Peter Kritzer
Johann Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, Austria

Examinateurs

Dr. Claire Mathieu
CNRS, Université Paris Cité, IRIF, France

Pr. Art B. Owen
Stanford University, United States of America

Dr. Julien Tierny
CNRS, Sorbonne Université, LIP6, France

Abstract

Discrepancy measures are metrics designed to quantify how well spread a point set is in a given space. Among these, the L_∞ star discrepancy is arguably one of the most popular. Indeed, by the Koksma-Hlawka inequality [Hla61; Kok43], when replacing an integral by the average of function evaluations in specific points, the error made is bounded by a product of two terms, one depending only on the function and the other on the L_∞ star discrepancy of the points. This leads to a variety of applications, from computer vision to financial mathematics and to design of experiments where well-spread points covering a space are essential.

Low-discrepancy sets used in such applications usually correspond to number theoretic designs, with a wide variety of possible constructions. Despite the high demand in practice, the design of these point sets remains largely the work of mathematicians, often more interested in finding asymptotic bounds than in adapting the point sets to the desired applications. This results in point sets that, while theoretically excellent, sometimes leave a lot to be desired for applications, in particular high-dimensional ones. Indeed, the constructions are not tailored to the many different settings found in applications and are thus suboptimal. Furthermore, not only do we not know how low the discrepancy of point sets of a given size in a fixed dimension can go, but often we do not even know the discrepancy of existing constructions. This leaves essential questions unanswered in the design of low-discrepancy sets and sequences.

In this thesis, we tackle the problem of constructing low-discrepancy sets from a computational perspective. With optimization approaches applied in isolation or on top of existing sets and sequences, we provide a diverse set of methods to generate excellent low-discrepancy sets, largely outperforming the discrepancy of known constructions in a wide variety of contexts. In particular, we describe a number of examples such as provably optimal sets for very few points in dimension 2, or improved sets of hundreds of points in moderate dimensions via subset selection. Finally, we extend recent work on greedy one-dimensional sequence construction to show that greedy L_2 construction of point sets provides excellent empirical results with respect to the L_∞ star discrepancy.

Résumé

Les mesures de discrépance sont une famille de mesures quantifiant l'uniformité d'un ensemble de points: sont-ils bien répartis dans l'espace (discrépance faible) ou concentrés par endroits (discrépance élevée)? Parmi ces mesures, la discrépance à l'origine L_∞ joue un rôle primordial: elle permet de quantifier l'erreur faite lors de l'approximation d'une intégrale par une somme finie composée d'évaluations de la fonction en un ensemble de points. Ce résultat mène à de très nombreuses applications dans des domaines variés allant de la vision en informatique à la finance, sans oublier tout ce qui concerne le plan d'expérience (design of experiments), où des points bien répartis sont essentiels.

Les points à faible discrépance utilisés dans ces applications sont issus d'approches mathématiques en théorie des nombres, avec de nombreuses séquences ou ensembles de points connus depuis plusieurs décennies. Malgré cela, ce sujet reste surtout le domaine de mathématiciens, souvent plus intéressés par les propriétés asymptotiques de ces points que leur utilisations en pratique. Il en résulte que les ensembles à faible discrépance, bien que très bons théoriquement, sont parfois moins bien que des points aléatoires pour certaines applications, particulièrement lorsque la dimension augmente. D'un point de vue plus qualitatif, malgré certaines bornes asymptotiques, nous ne savons pas précisemment quelle est la limite inférieure pour la discrépance d'un ensemble de points d'une taille fixée. Or, ce type de question a un impact immédiat sur la qualité des résultats lors d'applications.

Dans cette thèse, nous abordons le problème de la construction d'ensembles à faible discrépance d'un point de vue informatique, tout en obtenant de nombreux résultats empiriques sur la qualité de ces ensembles. Plutôt que de construire mathématiquement ces séquences, nous utilisons un ensemble de méthodes issues de l'optimisation, ainsi que les séquences existantes, pour obtenir de nouveaux ensembles de points. Nous montrons qu'il est possible d'obtenir des ensembles de points de bien meilleure discrépance à l'origine L_∞ que ceux existant déjà, dans des dimensions variées. Enfin, nous présentons également une approche basée sur la discrépance à l'origine L_2, pour montrer que son utilisation de manière gloutonne permet la construction d'une séquence extrêmement régulière pour la discrépance L_∞, ouvrant la voie vers de nombreuses approches novatrices.

Acknowledgments

I have immensely appreciated the three years at LIP6 working on this PhD. A large part of this is thanks to all the wonderful people at the team, who have come and gone in the last three years. It has been a pleasure coming to the office every day thanks to them. This thesis would not have been possible without the support and guidance of Carola Doerr and Luís Paquete. I am very grateful to them both for the scientific help and, from a more human side, the great atmosphere they created which has led me to enjoy so much this thesis. I would also like to extend my thanks to all the people I have met, discussed and worked with, whether in seminars, workshops or conferences. Large parts of this thesis would not exist without their contribution. These thanks naturally extend to the jury members, whether reviewers or examiners, who dedicated some of their valuable time to my work. Finally, I am very grateful to my family, that has always supported me and helped extensively on a myriad of non-scientific problems.

Contents

1 Introduction

1.1 Why Discrepancy?

Any first-year computer science student will have had the pleasure of being introduced to Monte Carlo methods via the approximation of π. The approximated value is obtained by randomly sampling from $[0,1]^2$, finding how many lie in the quarter-circle of radius 1 centered in the origin, and comparing this ratio to the ratio of the area of the quarter circle, $\pi/4$, and the total area, one, to obtain the value of π. In this use of a Monte Carlo method, one replaces the calculation of an integral by a finite sum over values of this function at specific points: the randomly chosen points.

This problem of approximating a continuous object by a discrete one lies at the heart of discrepancy theory. Is it possible to select the points better than randomly? It will come as no surprise to most readers that the answer is a resounding "yes" in the vast majority of cases. Over the past century, starting with Weyl's work in [Wey16], mathematicians have focused extensively on finding answers to the following questions:

1. By the central limit theorem, the error made when calculating an integral such as our π approximation is inversely proportional to the square root of the number of points sampled. Can we obtain better bounds with new sets?

2. How regular can these point sets be? Is it possible, for a fixed finite number of points, to get arbitrarily close to the uniform distribution?

3. How can one construct the most regular sets?

The second question is at the root of discrepancy theory. The different discrepancy measures give ways of measuring what "close" is. While there exist many different discrepancy measures to answer this, the key idea remains the same. For a point set P, its discrepancy is linked to the absolute difference between the measure of some subset $B \subseteq [0,1)^d$ and the proportion of points in P that fall inside B. Depending on the discrepancy measure, one can then consider the supremum or an integral over all possible subsets B. Some commonly considered families of subsets B are anchored boxes $\{[0,q) : q \in [0,1)^d\}$, axis-aligned rectangles $\{[a,b) : a,b \in [0,1)^d\}$, or simply convex or measurable sets.

An answer to the first question stated above reveals why these measures became prevalent. Separate results by Koksma [Kok43] and Hlawka [Hla61] led to the Koksma-Hlawka inequality: the bound on the error made when approximating a continuous integral by a discrete sum of function values over a set of points P directly depends on the discrepancy of P. Indeed, regardless of the choice of the function, the error will consist of the product of two separate terms, one based on the L_∞ star discrepancy of the points, and one determined by the function itself. Minimizing the discrepancy of the points therefore becomes a key objective when designing the sets used for numerical integration. In particular, we will see in more detail in Chapter 2 that we have numerous constructions whose discrepancy scales as $O(\log^{d-1}(n)/n)$,[1] whereas the discrepancy of random points for traditional Monte Carlo methods scales only in $O(\sqrt{d/n})$. Apart from specific settings,[2] one is thus better off using these regular constructions, *low-discrepancy sets and sequences*, than the usual random points. Essentially, any application that requires numerically computing an integral should use low-discrepancy point sets as long as d is not too large compared to n. This is the case for applications in computer vision [Pau+22] or financial mathematics [GJ97], for example. The error bound provided by the Koksma-Hlawka inequality results led to the development of a whole field called Quasi-Monte Carlo integration [DP10].

Quasi-Monte Carlo integration is not the only use of low-discrepancy point sets. Consider you are given a hidden function $f : [0, 1)^d \to \mathbb{R}$ and your goal is to find the maximum of this function. How do you choose which values to try? While random values or more methodical grid-based approaches could work, they will not perform as well as picking as your set of search points a low-discrepancy set, or sequence if you do not want a fixed budget. By being as uniformly distributed as possible, these sets cover the search space very well. The second major use of low-discrepancy sets is therefore simply when you want to find something in a search space, and desire better results than with random points. This leads to many applications, most of which can be grouped in the field of design of experiments [SWN03], for example one-shot optimization [Bou+17; Cau+20] or genetic algorithm initialization [MMM04].

1.2 Objectives of the Thesis

Despite their importance, there remain key practical and theoretical questions around low-discrepancy sets and sequences. As will be described in more detail

1 d is the dimension and n the number of points.
2 High dimension d and low number of points n.

in Chapter 2, the optimal order of discrepancy is generally not known, whether asymptotically for a sequence or for a fixed number of points. For a low number of points, the best discrepancy value that can be obtained is also unknown, even in dimension 2. Furthermore, from a practical perspective, while low-discrepancy constructions give a theoretical guarantee, the design is not tailored to an experiment. The gap between the mathematics community and those actively using the point sets also leads to relatively natural questions, such as how to optimally add a point to an existing set with respect to the discrepancy, to remain open.

While traditional approaches have relied on number-theoretical methods or asymptotic studies, the aim of this thesis is to use computational approaches to obtain excellent point sets for specific n and d. This is a much more ***user-focused*** perspective: for applications, there is only a finite budget, and thus finite number of points that can be used, making asymptotically good point sets a lot less valuable. In this thesis, we tackle the following question : **given a fixed number of points and dimension, what is the minimal discrepancy value one can obtain for a *constructible* point set P?**

We are therefore interested both in the problem of **designing new low-discrepancy sets**, and in that of **minimizing their discrepancy**.

While the focus will be on constructing low-discrepancy sets using optimization tools, we will also provide solutions to questions that may interest practitioners via methods such as subset selection, or greedily adding one or more points to an existing set. The goal is for practitioners to be able to modify low-discrepancy sets according to the needs of their applications while, hopefully, improving known constructions.

1.3 Outline of the Thesis

The thesis is split into two parts. In Part I, we introduce all the notions necessary to understand our work along with past results that either precede or help to contextualize our work. Chapter 2 presents the L_∞ star discrepancy and the key questions in the field. This covers topics from asymptotic bounds to low-discrepancy constructions, while highlighting where there are open questions relevant to our work. Chapter 3 describes the L_∞ star discrepancy from a computational perspective: how can it be computed and what are the main obstacles for its calculation? We also take the opportunity to describe a few very different approaches that were designed to evaluate the L_∞ star discrepancy. Finally, on a different note, Chapter 4

highlights some recent approaches based on greedy constructions, in particular in one dimension. We use this as an opportunity to describe the L_2 star discrepancy.

The contributions of this thesis are then found in Part II, going from the most precise algorithms in low dimensions to heuristics in higher dimensions with more points. We begin by describing methods to obtain point sets with provably optimal L_∞ star discrepancy values in dimension 2 and 3 in Chapter 5. A subset selection approach is introduced in Chapter 6, where we attempt to extract excellent low-discrepancy sets from existing constructions for moderate n in dimensions 2 and 3. We complement this with a short description of how to tackle the problem for the L_2 discrepancy. This is generalized to higher n in any dimension for which we can compute the star discrepancy with a heuristic approach in Chapter 7. Our last constructive approach in Chapter 8 describes how to greedily add a point to an existing set, with respect to the L_∞ star discrepancy or the L_2 star discrepancy. This last case corresponds to the so-called *Kritzinger sequence* [Kri22], which we extend to higher dimensions and a higher number of points. Finally, Chapter 9 and Chapter 10 underline the potential use of black-box optimizers in discrepancy, with two applications such as calculating the L_∞ star discrepancy and evaluating the periodic L_2 discrepancy of specific orthogonal stratifications of the unit cube.

We conclude this thesis with a presentation of future lines of research in Chapter 11 and some complementary numerical results in Chapter 12.

1.4 Contributions of the Thesis

The work presented in this thesis relies on the following publications and preprints, listed in order of appearance in the thesis. The ordering of the authors is alphabetical, except for the publication at GECCO 2023.

- **François Clément**, Carola Doerr, Kathrin Klamroth, Luís Paquete. Constructing Optimal Star Discrepancy Sets. A short version is under submission, while the full version is available on arxiv [Clé+23a].

- **François Clément**, Carola Doerr, Luís Paquete. Star Discrepancy Subset Selection: Problem Formulation and Efficient Approaches for Low Dimensions. *Journal of Complexity*, volume 70, 2022 [CDP22].

- **François Clément**, Carola Doerr, Luís Paquete. Heuristic Approaches to Obtain Low-Discrepancy Point Sets via Subset Selection. *Journal of Complexity*, volume 83, 2024. [CDP24]

- **François Clément**. Extending the Kritzinger Sequence: More Points and Higher Dimensions. Preprint. [Clé23].
- **François Clément**, Diederick Vermetten, Jacob de Nobel, Alexandre D. Jesus Carola Doerr, Luís Paquete. Computing Star Discrepancies with Numerical Black-Box Optimization Algorithms. *Proceedings of GECCO 2023*, pages 1330-1338, 2023 [Clé+23c].
- **François Clément**, Nathan Kirk, Florian Pausinger. Partitions for Stratified Sampling. *Monte Carlo Methods and Applications*, 2024 [CKP24].

We give a brief overview of these works, in the order of appearance in this thesis.

1.4.1 Constructing Optimal Star Discrepancy Sets

Constructing optimal point sets for the L_∞ star discrepancy is seen as a very hard problem in the discrepancy community. Indeed, provably optimal point sets were known only for $n \leq 6$ in dimension 2 and $n \leq 2$ for higher dimensions. In particular, there has been no progress since the 1970's and a paper by White [Whi77] for dimension 2.

In this chapter, we introduce mathematical programming formulations to construct point sets with as low L_∞ star discrepancy as possible. Firstly, we present two models to construct optimal sets and show that there always exist optimal sets in dimension 2 with the property that no two points share a coordinate. We then provide an extension of our models to higher dimensions, as well as to other discrepancy measures, for example the extreme and periodic discrepancies. For the L_∞ star discrepancy, we are able to compute optimal point sets for up to 21 points in dimension 2 and for up to 8 points in dimension 3. For dimension 2 and $n \geq 7$ points, these point sets have around a 50% lower discrepancy than the previously best known point sets. Furthermore, by plotting the local discrepancy values induced by the optimal sets, we observe a clear structural difference with that of known low-discrepancy sets, an example of which can be seen in Figure 1.1.

We also show that adding some regularity to our sets can be done at little cost. We introduce the *multiple-corner discrepancy*, a discrepancy measure close to the L_∞ star discrepancy that reduces the dependency on the corner in 0, by taking the maximum over all the star discrepancies anchored in each corner of $[0, 1)^d$. We show that optimal sets for this measure are not much worse for the L_∞ star discrepancy than those for the traditional L_∞ star discrepancy.

As mentioned in more detail in Chapter 11, all these elements combine to paint a very promising picture for future steps, with a large variety of approaches possible

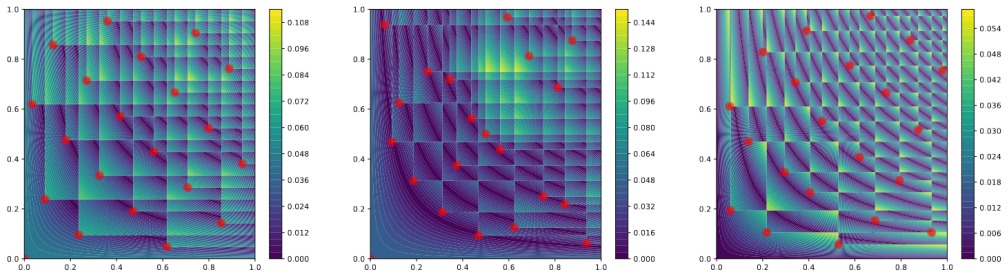

Figure 1.1: Fibonacci, Sobol' and optimal sets' local discrepancies for $n = 21$ in dimension 2. The colors are scaled with the maximum discrepancy value separately in each image. We observe that for the previous constructions, Fibonacci and Sobol', the sets are not well distributed because boxes very often contain too many points. The optimal set on the right presents a much more regular structure, with boxes with too many or too few points coming close to the worst discrepancy value.

to increase the number of points and the dimension, or to add multi-objective approaches to optimize not only the L_∞ star discrepancy but also other chosen measures.[3] This should be seen as a first step towards new constructions, and not simply better discrepancy values in specific cases.

1.4.2 Star Discrepancy Subset Selection: Problem Formulation and Efficient Approaches for Low Dimensions

Motivated by a problem of choosing diverse instances for training ML-based methods [Neu+18], we introduce the *star discrepancy subset selection problem*, which consists in finding a subset of k out of n points that minimizes the star discrepancy. First, we show that this problem is NP-hard. Then, we introduce a mixed integer linear formulation (MILP) and a combinatorial branch-and-bound (BB) algorithm for the star discrepancy subset selection problem. Both approaches are evaluated against random subset selection and a greedy construction on different use-cases in dimensions two and three, as well as the original low-discrepancy sets of size k. We note that both of these methods heavily rely on the grid structure introduced in Section 3.1 and apparent in Figure 1.1. Our results show that the MILP and BB are efficient in dimension two for large and small k/n ratio, respectively, and for not too large n. However, the performance of both approaches decays strongly for larger dimensions and set sizes.

3 This could be anything, from other discrepancy measures to geometric aspects such as the dispersion of the point set.

The point sets generated by this method have a much lower discrepancy value than nearly all previously known low-discrepancy sets and sequences of similar size. In particular, we observe that removing 20 points is often sufficient to greatly improve the discrepancy of the initial set. Though our results in this chapter are limited to dimensions 2 and 3, this suggests that subset selection could be an interesting approach for generating point sets of low dimension.

1.4.3 Heuristic Approaches to Obtain Low-Discrepancy Point Sets via Subset Selection

Building upon the exact methods presented in the previous chapter, we introduce a heuristic approach for the star discrepancy subset selection problem. The heuristic gradually improves the current-best subset by replacing one of its elements at a time, with a careful choice of the replaced point. While the heuristic does not necessarily return an optimal solution, we obtain very promising results for all tested dimensions. For example, for moderate sizes $30 \leq n \leq 240$, we obtain point sets in dimension 6 with L_∞ star discrepancy up to 35% better than that of the first n points of the Sobol' sequence. Our heuristic works in all dimensions, the main limitation being the precision of the discrepancy calculation algorithms.

We provide a comparison with a recent energy functional introduced by Steinerberger [Ste19][4], showing that our heuristic performs better on all tested instances. Finally, testing our heuristic led to numerous discrepancy evaluations, both for our newly-generated sets and for the Sobol' sequence. These give further empirical information on conjectures about the *inverse star discrepancy*, which describes the minimum number of points in a given dimension to reach a fixed discrepancy value.

1.4.4 Extending the Kritzinger Sequence: More Points and Higher Dimensions

This chapter is split into two parts, the second of which corresponds to [Clé23]. In the first part, we describe exact methods to greedily add a point optimally to a point set with respect to the L_∞ star discrepancy and show that repeating this method leads to a relatively poor sequence.

In the second part, we extend numerical experiments on the Kritzinger sequence[5], for which the greedy addition is done with respect to the L_2 star discrepancy. This

4 See Chapter 4.
5 Introduced in Section 4.3.

sequence was recently introduced in [Kri22] and, perhaps surprisingly, seems to perform just as well as some of the best one-dimensional low-discrepancy sequences. Though there are some upper bounds on its asymptotic discrepancy [Ste24], they do not seem to match the sequence's behaviour over the first few thousand points. We present in this chapter a simplified algorithm to compute millions of points in one dimension and show that the behaviour of the sequence not only stays extremely regular but, more importantly, seems to improve compared to the Kronecker sequence with golden ratio and the Ostromoukhov sequence. Furthermore, we introduce a set of exact and approximate methods to compute this sequence in dimensions 2 and 3. We show that the exact methods are always competitive with the Sobol' sequence for the hundreds of points we can construct, while the approximate methods give low-discrepancy sequences, even for thousands of points in dimension 2.

This chapter highlights two important points. The first is to confirm the existence of a very different approach to construct low-discrepancy sets that has been discovered recently (see Chapter 4). The second is that the L_2 discrepancy may be a very relevant tool for constructing low discrepancy L_∞ sets, especially as it answers two weaknesses of the L_∞ star discrepancy: its computational cost and its poor performance in greedy methods.

1.4.5 Computing Star Discrepancies with Numerical Black-Box Optimization Algorithms

As mentioned above, evaluating the star discrepancy of a given set is an expensive procedure. In order to find alternative algorithms, we apply a diverse set of off-the-shelf black-box algorithms and evaluate their performance on this problem. This approach has a secondary objective in that, should they perform badly, this problem would be able to provide a very large test set of instances in both continuous and discrete settings for black-box optimizers.

We compare eight popular numerical black-box optimization algorithms on the L_∞ star discrepancy computation problem, using a wide set of instances in dimensions 2 to 15. We show that all used optimizers perform very badly on a large majority of the instances and that in many cases random search outperforms even the more sophisticated solvers. We suspect that state-of-the-art numerical black-box optimization techniques fail to capture the global structure of the problem, an important shortcoming that may guide their future development.

Finally, we also took this opportunity to reimplement the best exact algorithm known to compute the discrepancy from [DEM96] to allow for parallelization. This

new implementation provides a speedup of a factor between 7.4 and 17.4. This enables us to get exact discrepancy values for both a higher number of points and higher dimensions.

1.4.6 Partitions for Stratified Sampling

The final chapter describes our use of black-box optimizers for the problem of computing optimal orthogonal stratifications of the unit cube with respect to the expected L_2 discrepancy. Classical *jittered sampling* partitions $[0,1]^d$ into m^d cubes for a positive integer m and randomly places a point inside each of them, providing a point set of size $N = m^d$ with small discrepancy. In the first part of the paper, not presented in this thesis, Kirk and Pausinger come up with methods to compute equivolume stratifications of the unit cube where each strata is bounded by two hyperplanes orthogonal to the unit cube's main diagonal. In relation to other papers by Kiderlen and Pausinger [KP21; KP22], the question was then to find how well this stratification performed compared to other jittered and stratified sampling methods.

Since the construction can be narrowed down to a choice of hyperplanes intersecting orthogonally the diagonal, we use three different black-box optimizers to attempt to place optimally those hyperplanes via their intersection points. Despite a very noisy setting, all three optimizers are able to obtain optimal sets of similar quality, better than the equivolume stratification for a low number of points. In particular, the optimal structure is quite unexpected, with an alternance of clusters of hyperplanes placed quite close together, and larger intervals with a single big strata. The extremal strata, closer to 0 and 1, are also usually smaller than in the equivolume case.

Part I

Background

2 Discrepancy Theory

Chapters 2 to 4 provide an introduction to discrepancy theory and, more importantly, to the key results underpinning this thesis. Content in these chapters may come from introductions in my papers, or be inspired by the books [Cha00; CST14; DGW14; Mat10]. In this first chapter, we introduce more formally discrepancy theory and in particular the L_∞ star discrepancy, *the* central notion of this thesis. We highlight known theoretical bounds and provide some well-known low-discrepancy constructions.

Throughout the thesis, n corresponds to the number of points in a point set $P := \{x^{(i)} : i \in \{1, \dots, n\}\}$, d is the dimension and we write $x^{(i)} = (x_1^{(i)}, \dots, x_d^{(i)})$, i.e., $x_j^{(i)}$ refers to the j-th *coordinate* of $x^{(i)}$. The only exception to this last notation will be for some of the mathematical programming formulations, and will always be explicitly mentioned.

2.1 The L_∞ Star Discrepancy

While there exist a vast set of discrepancy measures, this thesis focuses on the L_∞ star discrepancy. The L_∞ *star discrepancy* of a finite point *set* $P \subseteq [0,1)^d$ measures the worst absolute difference between the Lebesgue measure of a d-dimensional box $[0, q)$ anchored in $(0, \dots, 0)$ and the proportion $|P \cap [0, q)|/|P|$ of points that fall inside this box. More formally, it is defined by

$$d_\infty^*(P) := \sup_{q \in [0,1)^d} \left| \frac{|P \cap [0, q)|}{|P|} - \lambda(q) \right|. \tag{2.1}$$

One can see that *star* refers to the boxes anchored in 0, while the L_∞ comes from the supremum over all boxes. We call the term on the right inside the supremum *local discrepancy*, which we denote by $D_{\text{loc}}(q, P)$. Unless specified otherwise, "discrepancy", or "star discrepancy", refers to the L_∞ star discrepancy. Figure 2.1 gives an example of the local discrepancy for a specific box.

Star discrepancies defined with respect to the L_p norm are also well-studied. For

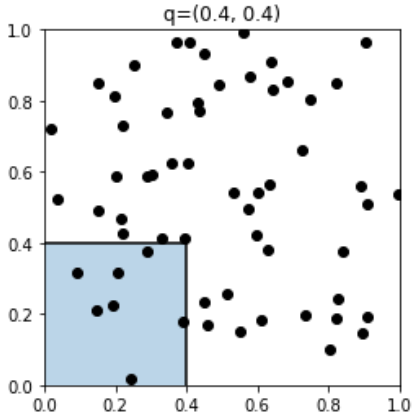

Figure 2.1: The local discrepancy in the box $[(0,0),(0.4,0.4))$ for a 60-point set. Here $D_{\text{loc}}(q,P) = 0.04333$ as the box contains 7 points and has volume 0.16. Shifting the top-right corner all over $[0,1)^2$ and computing the local discrepancy values would give us $d^*_\infty(P)$, the L_∞ star discrepancy of P.

a point set P, the L_p star discrepancy is defined by

$$d^*_p(P) := \left(\int_{[0,1)^d} \left(\left| \frac{|P \cap [0,q)|}{|P|} - \lambda(q) \right| \right)^p dq \right)^{1/p}. \tag{2.2}$$

In this thesis, out of the different L_p discrepancies, we only consider the L_2 discrepancy. Its main advantage is the ease with which it can be computed, despite some known drawbacks [Mat10]. It is described in more detail in Chapter 4.

Other discrepancy measures generally change the set of boxes used in the definition. While we use boxes anchored in 0 for star discrepancies, this is not systematically the case. For example, the *extreme discrepancy* considers all boxes in $[0,1)^d$ and the *periodic discrepancy* all boxes over $[0,1)^d$ considered as a torus. These are only mentioned in Chapter 5 and are therefore introduced in more detail there.

The previous setting with a fixed point set P should be distinguished from that of an infinite *sequence* of points $P' := (p^{(i)})_{i\in\mathbb{N}}$. One should now study the discrepancy depending on the number of points of the sequence one considers, given by the following function

$$d^*_\infty(P',k) := \sup_{q\in[0,1)^d} \left| \frac{|P'_k \cap [0,q)|}{k} - \lambda(q) \right|, \tag{2.3}$$

where P'_k corresponds to the k first elements of the infinite sequence P'. We are

interested in how this function evolves with k. In other words, we are not focusing on the discrepancy of a single set but in the worst discrepancy of an increasing sequence of nested sets and how to bound it.

While the work in this thesis tackles the problem of constructing low-discrepancy *sets*, the relationship between the two is close (see Section 2.2.1) and we use sets extracted from sequences as a comparison baseline on multiple occasions.

2.2 Theoretical Bounds

2.2.1 Asymptotic Orders

There are a vast number of settings possible in which we can consider discrepancy, depending on the dimension, set/sequence distinction, or type of point set. We begin by the settings in which the optimal order for the L_∞ star discrepancy is known, that is dimension 1 and sets in dimension 2. The easiest setting, and the only completely solved, corresponds to sets in dimension 1. The optimal discrepancy for an n-point set in dimension 1 is $d^*(n,1) = 1/(2n)$ and is obtained by the set $P := \{(2i+1)/2n : i \in \{0,\dots,n-1\}\}$. While it may have been known before, the first reference to this result is by Niederreiter in [Nie72].

For sequences in dimension 1, the optimal asymptotic order is $\Theta(\log(n)/n)$. Solving this problem corresponds exactly to the initial problem of discrepancy theory: does there exist a sequence of points $(x_n)_{n\in\mathbb{N}} \in [0,1)^{\mathbb{N}}$ such that for any q in $[0,1)$, the absolute difference between q and the proportion of points that fall inside $[0,q)$ is bounded by a constant divided by n? This was answered negatively by van Aardenne-Ehrenfest in [Aar49], and qualitatively by Roth in [Rot54]. Finding the optimal constant C in the $C\log(n)/n$ bound remains an open problem, the best asymptotic constant for a known sequence corresponds to a permutation of the van der Corput sequence by Ostromoukhov [Ost09] and is roughly equal to 0.22223. The best theoretical lower bound was found by Larcher and Puchhammer in [LP16] and is equal to 0.065.

For sets in dimension 2, a seminal result by Schmidt [Sch72] showed that the best possible discrepancy is $O(\log(n)/n)$. The fact that this matches the bound for sequences in dimension 1 is no coincidence. Indeed, it was shown by Roth in [Rot54] that the optimal order of discrepancy for a sequence in dimension d is the same as that of a set in dimension $d+1$, up to a constant factor. Of particular interest to us is the relation allowing us to *lift* a sequence P' in dimension d to a set P in dimension $d+1$ by setting $P := \{(x^{(i)}, i/n) : i \in \{1,\dots,n\}\}$ with the $x^{(i)}$

the first n points of the sequence.[6] The optimal constants are also not known, but this question for sets has attracted less interest than the closely related and more natural one-dimensional sequence problem.

In higher dimensions, finding the optimal order for the L_∞ star discrepancy is a wide open question and is referred to as the *Great Open Problem* [BC87]. For a sequence in dimension d, the upper bound is $O(\log(n)^d/n)$, and is reached by a large variety of constructions, see [Nie92] for a detailed description. Sets and sequences matching this bound are known as *low-discrepancy sets and sequences*. However, the best lower bound is $\Omega(\log^{d/2}(n)/n)$ by Bilyk, Lacey and Vagharshakyan [BLV08], and is the only progress made since Roth's initial $\Omega(\log(n)^{(d-1)/2})$ bound on the L_2 discrepancy in [Rot54], which is naturally a lower bound on the L_∞ star discrepancy.

A separate class of questions concerns random points and constructions such as Latin Hypercubes [MBC79]. The expected star discrepancy of a set P of n i.i.d. uniform random points in $[0,1]^d$ is of order $\Theta(\sqrt{d/n})$; see [Hei+01] for the first upper bound and [Ais11] for the first upper bound with explicit constant. The lower bound was obtained in [Doe14] , showing that these cannot match the optimal constructions.[7] Finally, [GPW20] gives the current state of the art results for the associated constant for random points. Completely random methods can be improved by adding some regularity to the construction, for example via jittered sampling: $[0,1)^d$ is decomposed into a d-dimensional grid with n boxes, and exactly one point per box will be placed. While this can only be defined for specific values of $n = m^d$, it does improve the expected discrepancy order to

$$E[d^*_\infty(P)] = \Theta\left(\frac{\sqrt{d}\sqrt{1+\log(n/d)}}{n^{\frac{1}{2}+\frac{1}{2d}}}\right), \tag{2.4}$$

as shown in [Doe22]. A first practical limitation of the best low-discrepancy sets and sequences is that while they have a better asymptotic order, as the dimension grows we will require a number of points exponential in d to keep the $\log^d(n)/n$ bound below $\sqrt{d/n}$, and even below 1. There is therefore no guarantee that they perform better than random points in such a context.

6 We note there exists a very distinct lifting method to go from a lower dimension set to a higher dimension one by Hinrichs and Oettershagen [HO14], potentially skipping more than one dimension. Despite its uniqueness and the many potential uses for such a method, it is not used in this thesis.

7 The same bounds also apply to Latin Hypercubes with randomly placed points in the selected boxes, provided that $d \geq 2$ and $n \geq 1\,600d$ [DDG18; GH21]. Note that in our experiments in Chapter 6, we deal with much smaller sample sizes and we use the "improved LHS" suggested in [BG02], for which the results do not immediately apply.

Finally, we note that finding the optimal order for the L_2 star discrepancy problem was solved much earlier, equal to $\Theta(\log(n)^{(d-1)/2})$. Davenport [Dav56] in dimension 2 and Roth [Rot79] in dimension 3 gave the first constructions before Chen and Skriganov gave a construction with asymptotic order in any dimension in [SC02].

2.2.2 Optimal Sets and the Inverse Star Discrepancy

While results on sets were already largely mentioned in the previous section, a major gap is visible: they only describe the asymptotic regime and not the best possible values for a given n. There are two related notions. Given a fixed n and d, the best possible star discrepancy of a point set of size n in dimension d is written as $d^*_\infty(n,d)$. Given a desired discrepancy ε and a dimension d, the smallest number of points $N^*_\infty(\varepsilon,d)$ required to reach $d^*_\infty(n,d) \leq \varepsilon$ is *the inverse star discrepancy*. It is known that $\min(\epsilon_0, cd/n) \leq d^*_\infty(n,d) \leq C\sqrt{d/n}$, where c, C and ϵ_0 are constants, thanks to work in [Hei+01] but, despite a variety of approaches in [DGS05; Doe+08; Gne08], no constructive solution is known. The upper bound naturally translates to the following bound on the inverse star discrepancy

$$N^*_\infty(\varepsilon,d) \leq \lceil C^2 d\varepsilon^{-2} \rceil.$$

Finding the relation between ϵ and d required to keep the inverse star discrepancy at a certain value is an open question [NW10],[8] as well as obtaining the best possible constant C [GPW20].

For fixed n and d such that n is not much larger than d, we have $d^*_\infty(P) \geq \min\{c_0, cd/n\}$, where $c_0, c \in (0,1]$ are suitable constants [Hin04]. On the positive side, there exist n-point sets P such that $d^*_\infty(P) \leq C\sqrt{d/n}$, for some universal constant $C > 0$ [GH21; Hei+01].[9] Uniformly sampled i.i.d. points satisfy the upper bound in expectation and also with high probability [Doe14; Hei+01].

Nevertheless, there is a clear lack of meaningful results for the setting where n and d are of a similar order, in particular when n is very small. The only general lower bound that holds for nearly all n is by White [Whi77] in dimension 2 who showed that $d^*_\infty(P) \geq 1/n$ for any point set P of size at least 6 in dimension 2. A generalization of this result is shown in Chapter 5. Optimal constructions for $n \leq 6$ in dimension 2 were shown by White in that same paper. Pillard, Cools, and Vandewoestyne [PVC06] proved the optimal $n = 1$ sets for all dimensions for the L_∞ star discrepancy as well as the L_2 and extreme discrepancies. Larcher and Pillichshammer [LP07] then extended these results to $n = 2$ points. For the L_2

8 Or, equivalently, finding a better relation between n and d to bound $d^*(n,d)$.

9 This is clearly only required when $log^{d-1}(n)/n$ is close to 1.

periodic discrepancy, Hinrichs and Oettershagen [HO14] were able to use symmetry groups to reduce the problem space and obtain optimal sets for up to $n = 16$ points in dimension 2. None of the methods used in these papers can be easily used for higher n, or be generalized to other discrepancy measures in the last case.

This lack of results should not be seen as a lack of interest in the question, but more a sign of its difficulty from a mathematical perspective. Indeed, Novak and Wozniakowski [NW10] describe some open questions on the inverse star discrepancy in Open Question 43. While these were partially solved by de Rainville and Doerr in [DR13],[10] experimental results suggest much better is feasible.

This thesis provides a wide variety of numerical experiments to tackle the following problems.

- Optimal values for very small sets in dimension 2 and 3 are investigated in **Chapter 5**. This is complemented by possible directions for heuristics with a larger number of points.
- Conjectures on the inverse star discrepancy are studied in **Chapter 7** via evaluations of the Sobol' sequence and our subset selection results.
- The optimal constant for a one-dimensional sequence is examined via the construction of millions of points of the Kritzinger sequence in **Chapter 8**.

2.2.3 Upper-Bounding the Star Discrepancy

We finish this theoretical section with a quick description of some upper bounds on the star discrepancy. This is an especially important question for practitioners, as obtaining a lower bound is comparatively easy since one only needs to evaluate a local discrepancy. The main result is the Erdős-Turán-Koksma inequality [ET48a; ET48b; Kok50], which upper-bounds the star discrepancy of a point set P in any dimension

$$d^*_\infty(P) \lesssim_d \frac{1}{M+1} + \sum_{||k||_\infty \leq M} \frac{1}{r(k)n} \left| \sum_{j=1}^{n} e^{2\pi i \langle k, x^{(j)} \rangle} \right|, \tag{2.5}$$

where, for $k \in \mathbb{Z}^d$,

$$r(k) = \prod_{j=1}^{d} \max(1, |k_j|),$$

10 Their discrepancy values were computed with the TA heuristic (see Section 3.4), and are therefore not guaranteed to be exact.

and $a \lesssim_d b$ corresponds to $a \leq c(d)b$ where $c(d)$ is a constant depending on d. In one dimension, the Erdős-Turán inequality [ET48a; ET48b] provides a simpler bound.

$$d^*_\infty(P) \lesssim \frac{1}{n} + \sum_{k=1}^{n} \frac{1}{kn} \Big| \sum_{j=1}^{n} e^{2\pi i k x^{(j)}} \Big| \tag{2.6}$$

These inequalities are representatives of a bigger class, described in [DP10; Nie92]. Section 10.3.4 in [DGW14] describes numerical experiments performed to evaluate these bounds, as well as results from [Joe12; Nie92; SJ94]. In particular, results from [Joe12] show that for dimensions where we cannot compute the discrepancy, and would need these bounds, the obtained values are far greater than 1, a trivial bound for the star discrepancy. We have therefore not considered using these bounds, especially as their simpler forms for lattices cannot be used with our unstructured sets. Nevertheless, they are useful as a starting point for an energy formulation by Steinerberger in [Ste19]. A comparison with this method is done in Chapter 7, while energy functionals more generally are described in Chapter 4.

2.3 Well-Known Low-Discrepancy Sets and Sequences

We very briefly describe here some classic low-discrepancy constructions, whether sets or sequences. The description is largely taken from [CDP22] and the longer version of [Clé+23a] with some added explanations. A more detailed description can be found in the books by Niederreiter [Nie92] or Dick and Pillichshammer [DP10]. This should not be seen as an exhaustive list, but merely the sets that were used in the following chapters of this thesis.

- **Sobol' sequences** [Sob67], also called (t, d)-sequences in base 2: For two integers $0 \leq t \leq m$, a *(t, m, d)-net in base b* is a set of points $P = \{x^{(1)}, \dots, x^{(b^m)}\}$ such that for all "elementary" boxes I of the form $\prod_{j=1}^{d} [\frac{a_j}{b^{d_j}}, \frac{a_j+1}{b^{d_j}})$, with $a_j, b \in \mathbf{N}$ satisfying $0 < a_j < b^{d_j}$, and volume $\lambda(I) = b^{t-m}$ it holds that $|I \cap P| = b^t$. For $t \in \mathbf{N}$, a *(t, d)-sequence in base b* is a sequence of points $(x^{(i)})_{i \in \mathbf{N}}$ such that for all integers $k > 0$ and $m \geq t$ the set $\{x^{(kb^m)}, \dots, x^{((k+1)b^m-1)}\}$ is a (t, m, d)-net in base b. (t, m, d)-nets are particularly important as they guarantee precise discrepancy values when $n = b^m$. These then allow us to bound the discrepancy for all other intervals using (t, d)-sequences' properties. Various ways to construct Sobol' sequences exist. The most efficient techniques use Gray code representations of integers.

Sobol' sequences differ in the initialization numbers, and several works exist, which list good initialization for different dimensions, see [JK08] for examples, references, and implementations. An example using the GNU Scientific Library in dimension 2 is given in Figure 2.2 (left).

- **Faure sequence** [Fau82] is a $(0, d)$-sequence using as prime base the smallest prime number b satisfying $b \geq d$.

- **Halton sequence** [Hal64]: Let $b_1, \ldots, b_d > 1$ be co-prime numbers. Define the sequence $P = (x^{(i)})_{i \in \mathbb{N}}$ by setting, for each $j \in \{1, \ldots, d\}$, $x_j^{(i)} := \sum_{k \geq 0} d_{j,k}(i)/b_j^{k+1}$, where $(d_{j,k}(i))_{k \in \mathbb{N}}$ is defined as the unique sequence of integers $0 \leq d_{j,k}(i) < b_j$ such that $i = \sum_{k \geq 0} d_{j,k}(i) b_j^k$. That is, $(d_{j,k}(i))_{k \in \mathbb{N}}$ is the b_j-ary representation (also known as b_j-*adic expansion*) of i, and the Halton points "inverses" this representation to obtain numbers in $[0, 1]$. The first 128 points of the sequence in dimension 2 are given in Figure 2.2 (center). This can be generalized by allowing for permutations of the elements for each term of the sum. We then have $x_j^{(i)} := \sum_{k \geq 0} \pi_k(d_{j,k}(i))/b_j^{k+1}$, where the π_k are permutations of $\{0, \ldots, b-1\}$. The Halton sequence is naturally obtained with all the π_k being the identity.

- The Halton sequence is a generalization of the older **van der Corput sequence** [Cor35], which is defined in dimension 1. Apart from the traditional definition, finding good permutations of the elements in the generalized reverse binary sums has received much attention, well summarized in [Pau19]. The **Ostromoukhov** construction [Ost09] giving the to-this-day best upper bound for the constant in the discrepancy order is one of these constructions. We point out in particular that while this sequence is not competitive with the Fibonacci sequence below for all n, it is excellent when n is a power of 2.

- **Reverse Halton sequence**: It is known that Halton sequences show some unwanted correlations in the two-dimensional projections (unless the dimension d is very small), see [DGW14] for an example. To address this shortcoming, different scrambled versions have been suggested. In our experiments we use the RevHal constructions suggested in [VC06].

Finally, a common construction for low-discrepancy sets is to build lattices $\{zi/n : z \in \mathbb{N}^d, i \in \{0, \ldots, n-1\}\}$, separately introduced by Korobov and Hlawka (see for example [Hla62; Kor59]). Choosing the parameter z well is a difficult problem [Nie92, Section 5]. A typical choice is usually integers $z = (z_1, \ldots, z_d)$ with no factor in common with n. These sets are particularly important in practice as they

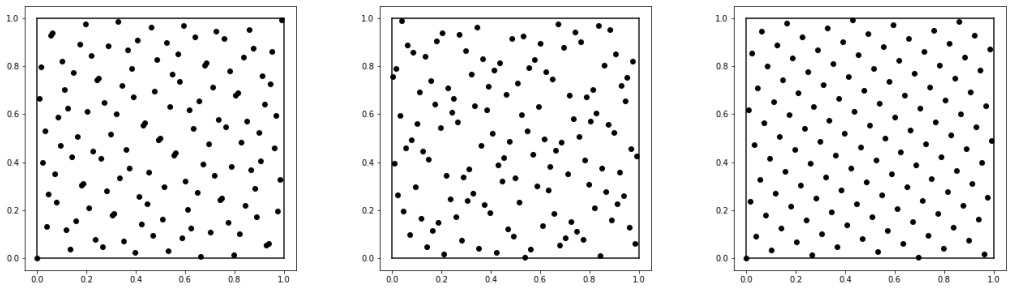

Figure 2.2: The first 128 points of the Sobol' sequence (left), Halton sequence (center) and Fibonacci set (right). The discrepancy of the Sobol' points is 0.024781, that of the Halton points is 0.036169 and 0.020254 for the Fibonacci set. It should not be forgotten that for sequences we expect worse discrepancy values as they need to be have low discrepancy for all n, while the Fibonacci set would need to be recomputed for other n.

allow to exploit increasing smoothness of a function when trying to numerically calculate an integral. They have been very extensively studied, both with integer parameters or more generally with elements of $\mathbb{R}^s/\mathbb{Z}^s$ (general lattice rules, which is closer to what we obtain here). In particular, choosing rationals z_j/n with small continued fraction expansion leads to good lattices (we refer once again to the book by Niederreiter [Nie92]). Setting aside integer parameters, this continued fraction expansion property is reflected by the excellent performance of the **Kronecker sequence with golden ratio**, which we will also refer to as the **Fibonacci sequence**. It is defined by $(\{n\phi\})_{n\in\mathbb{N}}$, where ϕ is the golden ratio[11] and $\{n\phi\}$ is the fractional part of $n\phi$. This can be extended to a 2-dimensional set of lattice form also called **Fibonacci set**: $\big\{(i/n, \{i\phi\}) : i \in \{0, \dots, n-1\}\big\}$. The Fibonacci set for $n = 128$ is given in Figure 2.2 (right).

While general lattices do not appear much in this thesis, this set plays an important role in Chapters 5 and 6, and as a sequence in Chapter 8 as it is the one with lowest discrepancy in dimension 1.[12]

In addition to these low-discrepancy sets and sequences, we also refer to two random constructions in Chapter 6, uniform sampling and Latin Hypercubes. While the former can be seen as a sequence, the latter is always a set.

11 $\phi = (1 + \sqrt{5})/2$.

12 This is an empirical remark. The best theoretical constant is obtained by Ostromoukhov with a van der Corput sequence permutation.

- **Uniform sampling**: We simply select $x^{(i)} \in [0,1]^d$ uniformly at random, and do this independently for each i.

- **Improved Latin Hypercube Sampling**: Classical Latin Hypercube sampling requires to sample d permutations $\sigma_1, \ldots, \sigma_d$ of the set $\{1, \ldots, n\}$ and to set $x_j^{(i)} := (\sigma_j(i) - u_j^{(i)})/n$, where $0 \leq u_j^{(i)} < 1$ denotes a uniformly sampled value. That is, we select the i-th point $x^{(i)}$ by choosing it randomly in the box $[(\sigma_j(i) - 1)/n, \sigma_j(i)/n]^d$. The advantage of LHS over uniformly selected Monte Carlo points is that the one-dimensional projections are all well spread. A disadvantage is that the points can nevertheless be close to each other, e.g., when σ_j is the identity permutation for all $j \in \{1, \ldots, d\}$ (in which case the n points are all close to the diagonal). Various versions of LHS have been suggested in the literature. We use the "improved" LHS construction suggested in [BG02]. This variant constructs the set P iteratively, by sampling at each stage a few alternatives and then selecting the candidate that maximizes the distance to the points that are already collected in the set P.

3 A Computational Perspective on Discrepancy

After this brief discovery of discrepancy theory, we turn our attention to computational aspects related to it. We first describe the grid structure of the L_∞ star discrepancy in Section 3.1, arguably the result on which this thesis relies the most. We then give a summarized explanation of the two main algorithms to compute the star discrepancy in Sections 3.3 and 3.4, before mentioning some past uses of optimization methods for discrepancy questions such as its computation in Sections 3.5 and 3.6.

3.1 The Grid Structure of the L_∞ Star Discrepancy

Despite being defined as a continuous problem over all possible anchored boxes, calculating the star discrepancy can be treated as a discrete problem [Nie72]. First, we notice that any closed anchored box in $[0,1]^d$ can be obtained as the limit of a sequence of bigger open boxes that contain the same number of points. The only exception is $[1,\dots,1]$ and this closed box cannot give the worst discrepancy value as its local discrepancy is 0. We define $D(q,P)$ to be the number of points of P that fall inside the open anchored box $[0,q)$ and $\overline{D}(q,P)$ the number of points of P that fall inside the closed anchored box $[0,q]$. We define the two following functions:

$$\delta(q,P) := \lambda(q) - \frac{1}{n}D(q,P) \qquad \text{and} \qquad \overline{\delta}(q,P) := \frac{1}{n}\overline{D}(q,P) - \lambda(q). \tag{3.1}$$

The local discrepancy $D_{\text{loc}}(q,P)$ in a point $q \in [0,1]^d$ is given by the maximum of $\delta(q,P)$ and $\overline{\delta}(q,P)$. It is not necessary to consider all possible values for q. Indeed, we can define for all $j \in \{1,\dots,d\}$ the grid

$$\Gamma(P) := \Gamma_1(P) \times \ldots \times \Gamma_d(P) \qquad \text{and} \qquad \overline{\Gamma}(P) := \overline{\Gamma}_1(P) \times \ldots \times \overline{\Gamma}_d(P), \tag{3.2}$$

with

$$\Gamma_j(P) := \{x_j^{(i)} | i \in 1,\dots,n\} \qquad \text{and} \qquad \overline{\Gamma}_j(P) := \Gamma_j(P) \cup \{1\}, \tag{3.3}$$

As shown in more detail in [DGW14], the star discrepancy computation reduces

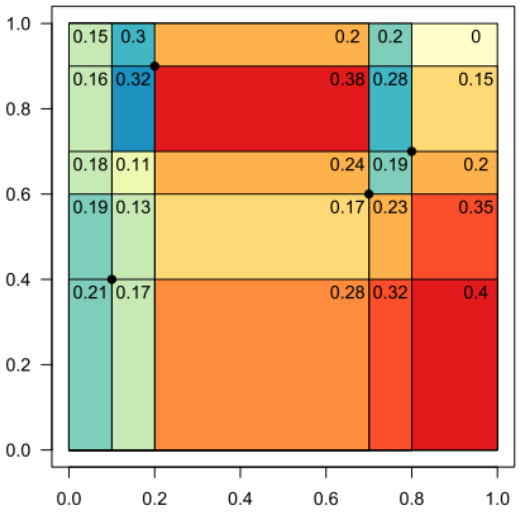

Figure 3.1: An illustration of the grid boxes defined by equation (3.4). The value indicated in a box corresponds to the discrepancy value in the top-right corner of this box. Blue boxes contain too many points and are closed, while red boxes contain too few and are open. This figure is taken from [CDP22].

to the following discrete problem

$$d_\infty^*(P) = \max\left\{ \max_{q\in\overline{\Gamma}(P)} \delta(q,P), \max_{q\in\Gamma(P)} \overline{\delta}(q,P) \right\}. \tag{3.4}$$

An illustration of the grid in this formula is provided in Figure 3.1.

Equation (3.4) allows us to ignore the absolute value in the definition of the star discrepancy in equation (2.1). If the sign of $|P \cap [0,q)|/n - \lambda([0,q))$ is positive, this corresponds to a box with too many points compared to its volume. This can be associated with the closed boxes in the second term of equation (3.4). If it is negative, this corresponds to an open box with too few points, the first term in equation (3.4). The distinction between open and closed is important: while a closed box can also have too few points, one would always obtain a worse value by considering the open box with the same top-right corner, and a similar comment is true when swapping the roles. One can safely assume in this thesis that an overfilled box is considered closed while an underfilled one is open.

The key argument behind this formula is that, for a box $[0,q)$ (or $[0,q]$), if the top-right corner is not in a grid position then it is always possible to shift q slightly to make the discrepancy worse. This argument can be used to reduce the set

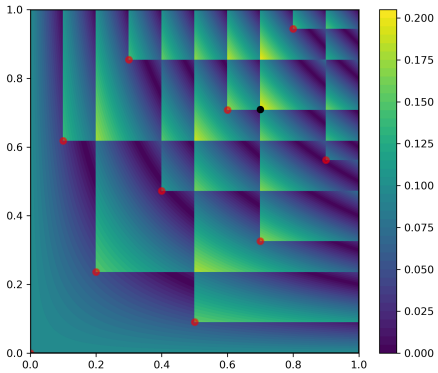

Figure 3.2: An illustration of critical boxes for a random set. One can see that each point defines a line going upwards and one to the right, where all grid points are located. The $1/n$ jumps in local discrepancy values when crossing these axes are also visible.

of candidate boxes even further. A box $[0, q)$ (or $[0, q]$) can obtain the maximal discrepancy value only if for all $i \in \{1, \dots, d\}$ there exists a point $p \in P$ such that $p \in [0, q]$ and $p_i = q_i$ (where we write $q = (q_1, \dots, q_d)$). These specific boxes $[0, q)$ and $[0, q]$ are called *critical boxes*. In Figure 3.2, the critical boxes are visible as the intersection points of the grid lines, and the points themselves. Critical boxes are defined for both open and closed boxes, the difference being that for open boxes the points defining its edges will not be inside the box. An explicit formulation of the discrepancy formula induced by these critical boxes in dimension 3 is given by Bundschuh and Zhu in [BZ93]. The lack of publications describing this formula should not be seen as a theoretical difficulty, but simply because notations would become cumbersome, without gaining theoretical insights or practical uses.

With equation (3.4), computing the star discrepancy becomes a discrete problem. A naïve enumeration of all possible solutions can be done in time $O(n^d)$. Restricting this search to critical boxes reduces the complexity to $O(n^d/d!)$. This is clearly too large to be done in any setting, in particular in higher dimensions. To our knowledge, this method has been used only up to around dimension 6 [WF97] and, as we will see in Section 3.3, it is not advisable to use it in dimensions higher than 2. Nevertheless, we heavily rely on the grid $\overline{\Gamma}(P)$ for our exact methods in Chapters 5 and 6, as we will do more than just compute the discrepancy.

3.2 Complexity of Calculating the L_∞ Star Discrepancy

Is there any hope in greatly improving the complexity given by the critical box enumeration? The short answer is no: any exact algorithm has a complexity such that a term of the shape $n^{f(d)}$ appears. It is impossible to eliminate this exponential dependency on d. Two main results lead to this conclusion.

The first is the proof of NP-hardness, and even NP-completeness, of calculating the L_∞ star discrepancy by Gnewuch, Srivastav and Winzen in [GSW09]. Their proof is based on a two-step reduction from Dominating Set. They first show it is possible to reduce Dominating Set to the problem of finding if there exists a box with volume at least ε that contains at most k points, and similarly to the problem of finding if there exists a box with volume at most ε containing at least k points. A second reduction is then made to obtain the NP-hardness of the L_∞ star discrepancy calculation. NP-completeness is a natural consequence as Dominating Set is an NP-complete problem and calculating the discrepancy is clearly in NP.[13]

The second, more important, result is an elegant proof by Giannopoulos, Knauer, Wahlström and Werner of the W[1]-hardness of calculating the L_∞ star discrepancy in [Gia+12]. Without going into the details of W-complexity classes,[14] they represent a growing set of complexity classes in parametrized complexity. "Easier" problems are *fixed-parameter tractable* and their complexity can be written in $f(d)n^{O(1)}$. For these, it is possible to separate the dependency between a parameter d in the problem and the problem's size n. This class includes all problems in P, as well as some in NP, such as optimization problems with an efficient polynomial time approximation scheme. The W[]-hierarchy represents a set of problems growing in difficulty, the "easiest" being in W[1]. A W[1]-hard problem has a complexity in $n^{O(d)}$, for example finding if a graph has a clique of size d. For the W[1]-hardness of the star discrepancy, the proof is done via a reduction of clique. $2d$ dimensions are considered, divided into pairs and in each a parabolic staircase of points will be built. These points alternate between points that can be selected and represent an element of the problem, and forbidden points that will prevent the selection of specific pairs of points at the same time. It is quite complicated and we defer to their paper for a thorough explanation.

One key element visible in both the NP-hardness and the W[1]-hardness proofs is that the point sets built are very specific: the worst discrepancy value is always very large, close to 1, and reached for the largest empty box or the smallest closed box. To

13 Given a box corner, one can check its local discrepancy naïvely in $\Theta(dn)$.
14 We refer an interested reader to [Cyg+15].

our knowledge, there are no complexity proofs that add some regularity conditions on the point set to make it more realistic. It is theoretically possible, though in our opinion very unlikely, that finding the discrepancy of low-discrepancy sets is an easier problem.

3.3 The Dobkin, Eppstein, Mitchell Algorithm

The best exact algorithm to compute the star discrepancy was introduced by Dobkin, Eppstein and Mitchell in [DEM96]. We will refer to it as *DEM algorithm* throughout this thesis, as we will rely on it consistently for all discrepancy calculations in low dimensions. It involves building a decomposition of $[0,1]^d$ in $O(n^{d/2})$ disjoint boxes B_i, such that we can find the maximum local discrepancy for a top-right corner in B_i in linear time. This leads to a total complexity of $O(n^{1+d/2})$. The algorithm can be used up to dimension 8 for a few hundred points or dimension 10 for a few dozens. While the initial algorithm in [DEM96] was based on a point-box dualization, our description in this subsection follows a more direct approach, as in Chapter 10 of [DGW14] and in the original implementation by the authors of [GWW12]. Some border cases are ignored for the clarity of the proof, as they do not change the general idea of the implementation, or the stated complexity.[15]

Firstly, a point x is said to be *internal* in dimension j for a box $[a,b]$ if $a_j < x_j < b_j$. Starting from $[0,1]^d$, we build dimension-by-dimension boxes $[a,b]$ such that the two following properties are verified:

1. Any $x \in [0,b)$ is internal in at most one dimension;
2. For each box built up to dimension j in $\{1,\dots,d\}$, there are at most $O(\sqrt{n})$ points internal in dimension j - for a fully-built box $[a,b]$ this holds for all dimensions.

The decomposition is built recursively in the following manner. We consider $[0,1]^d$ and start the decomposition in the first dimension. We find the smallest coordinate $c_{1,1}$ such that $[0,c_{1,1}] \times [0,1]^{d-1}$ contains $\sqrt{n}$ points, then $c_{1,2}$ such that $[c_{1,1},c_{1,2}] \times [0,1]^{d-1}$ contains $\sqrt{n}$ points and continue until we obtain a set $(c_{1,i})_{i\in\{0,\dots,\lceil\sqrt{n}\rceil\}}$ where $c_{1,0} = 0$ and the last non-zero $c_{1,j} = 1$. For each of these boxes $[c_{1,i},c_{1,i+1}] \times [0,1]^{d-1}$, we track which points are internal in dimension 1 (at most $\sqrt{n}$) and which points are inside the box $[0,c_{1,i+1}) \times [0,1]^{d-1}$.

15 The following description is necessary only for a reader interested in our parallelized implementation in Chapter 9. Understanding the algorithm is by no means necessary to read the rest of the thesis.

Given a box $B_j = [c_{1,i_1}, c_{1,i_1+1}] \times \ldots \times [c_{j,i_j}, c_{j,i_j+1}] \times [0,1]^{d-j}$, we recursively perform a similar decomposition in dimension $j+1$. Let n_j be the number of points inside $\overline{B_j} := [0, c_{1,i_1+1}) \times \ldots \times [0, c_{j,i_j+1}] \times [0,1]^{d-j}$. The new $c_{j+1,i}$ need to verify the two following properties. Firstly, for any point x in $\overline{B_j}$ that is internal in one of the first j dimensions, there needs to be some $c_{j+1,i} = x_{j+1}$. Secondly, for any $i \in \{0, \ldots, \lceil\sqrt{n}\rceil - 1\}$, there are at most $\sqrt{n}$ points y in $\overline{B_j}$ such that $c_{j+1,i} < y_{j+1} < c_{j+1,i+1}$. The first property guarantees that a point will never be internal in multiple dimensions, and the second that not too many points are in the boxes obtained from $\overline{B_j}$.

After d steps, we obtain boxes $[a,b]$ verifying the two desired properties: a point is internal in at most one dimension and there are at most $O(\sqrt{n})$ points internal in each dimension. We can now find the worst local discrepancy for a box whose top-right corner is in $[a,b)$ with the following dynamic programming approach. Let $m(h,j)$ (respectively $r(h,j)$) be the maximum (respectively minimum) value of $\prod_{i=1}^{j} y_i$ such that the box $[0,y_1) \times \ldots \times [0,y_j) \times [0,b_{j+1}) \times \ldots \times [0,b_d)$ contains exactly h points. By ordering the points in $[0,b]$ according to their first dimension, we can easily obtain $m(\cdot,1)$ and $r(\cdot,1)$. Since points can only be internal in a single dimension, taking an internal point in dimension 1 guarantees it will be contained in the box regardless of the other choices. From the $m(\cdot,j)$ and $r(\cdot,j)$, we can therefore calculate $m(\cdot,j+1)$ and $r(\cdot,j+1)$. Calculating each $m(\cdot,j+1)$ takes $O(\sqrt{n})$ time as there are $O(\sqrt{n})$ internal points and therefore potential different choices for the coordinate y_{j+1}. In total, there are at most $d\sqrt{n}$ internal points, $m(\cdot,j+1)$ needs to be computed for $O(d\sqrt{n})$ different values. The dynamic programming takes $O(dn)$ time (the d factor is usually ignored). All that remains to be done is to find $\max_{0 \le h \le n}(m(h,d) - h/n, h/n - r(h,d))$, in other words which (number of points inside a box, box volume) combination gives the worst discrepancy. This will directly give the worst discrepancy value for a box whose top-right corner is in $[a,b)$. Iterating over all boxes gives the stated complexity.

An improved parallelized version of this algorithm implemented during this thesis with A.D. Jesus can be found in Chapter 9.

3.4 Threshold Accepting

Several applications of point sets with low star discrepancy value concern settings that are not efficiently tractable by the exact algorithms described in the previous two sections. For these applications, we therefore need to resort to heuristic approaches to evaluate the discrepancy of a given point set. To date, the best-known heuristic is a Threshold Accepting algorithm proposed in [GWW12]. It

will be referred to as the *TA algorithm* or *TA heuristic* in this thesis. This approach exploits the grid structure introduced in Section 3.1 and operates on the search space $[1, \dots, n+1]^d$, with each point encoding one of the grid points in $\overline{\Gamma}(P)$. We will rely on it for all discrepancy calculations that cannot be done with the DEM algorithm.[16]

The TA algorithm from [GWW12] builds on an earlier approach by Winker and Fang suggested in [WF97] and extends it by adding various problem-specific components. Threshold Accepting [DS90] is similar to Simulated Annealing [KGV83] but replaces its probabilistic selection criterion with a deterministic one. That is, at each step, the current incumbent solution is compared to a randomly sampled neighboring solution. The neighbor is selected as the new center if its quality is not much worse than that of the previous incumbent. More precisely, neighbor y replaces incumbent x if $f(y) - f(x) \geq \tau(t)$, where $\tau(t) \leq 0$ is the threshold chosen at iteration t. The sequence $(\tau(t))_t$ is monotonically increasing so that the further advanced the optimization process is, the harder the selective pressure.

The TA algorithm from [GWW12] uses a dynamic choice of the neighborhood structure, increasing the number of coordinates $\{j \mid x_j \neq y_j\}$ that may change in each iteration while at the same time decreasing the absolute difference $|x_j - y_j|$. The more important problem-specific component, however, is a "snapping" routine, which rounds a selected grid point to a so-called critical box; see [GWW12] for details.

Using the DEM solver from Section 3.3 as a baseline, it was shown in [GWW12] that the TA algorithm successfully found the optimum on all instances for which DEM could provide exact values. Based on the results presented in [GWW12], the TA algorithm seems reliable for point sets up to dimensions 12 to 20 for a few hundred points. This imprecision should be kept in mind for Chapter 7 where some discrepancy values are computed in high dimensions.

3.5 Other Approximation Methods of the Star Discrepancy

While not directly used in this thesis, the methods presented in this section are structurally close to our work, whether it is via similar optimization methods in Thiémard's work or improving the grid $\Gamma(P)$ for bracketing covers. These have appeared regularly in our discussions during the thesis, and we hope that their

16 We only use the algorithm without modifying it in the thesis, as such the following description is not essential to understand our results from Chapter 5 onwards.

description here inspires future ideas. A reader interested solely in understanding the chapters covering our work may skip this section safely.

3.5.1 Bracketing Covers

Firstly, it is possible to approximate *additively* the L_∞ star discrepancy, via the use of bracketing covers. For our purpose, we define a bracketing δ-cover $\mathcal{B}$ as a set of boxes anchored in 0 such that, for any box $[0, q)$, there exist two boxes $l, u \in \mathcal{B}$ such that $q \in [u, l]$ and $||u - l|| \leq \delta$. In our context, bracketing covers were introduced by Dudley in [Dud78] and have been extensively studied, see for example the survey paper [Gne12]. The most recent improvement in the field is by Gnewuch in [Gne24], giving a new upper bound for the best possible size of a bracketing cover.

One can notice that with a fine enough bracketing cover[17] (i.e. for a point set P, no two parallel grid lines of $\Gamma(P)$ pass through $[u, l]$ if $||u - l|| \leq \delta$ and $[u, l]$ does not contain another point of the bracketing cover), for any box $[0, q)$, there exist two boxes u and l such that one contains as many points as $[0, q)$ and the other as many as $[0, q]$, while meeting the conditions of the δ-bracketing cover. Supposing $u \leq l$, the local discrepancy for $[0, q)$ is at most δ away from that of $[0, u]$ as it contains the same number of points and its volume is at most different by δ. A similar argument holds for $[0, l]$ and $[0, q]$. Therefore, computing the local discrepancy values over all elements of the bracketing cover guarantees an error of at most δ on the L_∞ star discrepancy computation. This result was first shown in [DGS05]. While bracketing covers could be useful to obtain a smaller grid than $\Gamma(P)$, they were not used in our work. The main reason for this is their size, that grows exponentially in the dimension: $d^d \delta^{-d}/d!$ in dimension greater or equal than 3 based on the most recent result in [Gne24], which improves the upper bound in the number of boxes created in a decomposition by Thiémard in [Thi01a]. Finally, there is no known multiplicative approximation algorithm for the L_∞ star discrepancy.

3.5.2 Thiémard's Optimization Method

Thiémard's papers [Thi01a; Thi01b] and thesis [Thi00] are one of two examples of the use of optimization methods in discrepancy, the other being by de Rainville and Doerr in [DR13] in the next section.

17 This is only for ease of exposition. Having a δ-cover suffices.

Foreshadowing the NP-hardness proof in [GSW09], Thiémard splits the discrepancy calculation problem into $2n$ different problems in [Thi00; Thi01b]: what is the largest/smallest volume a box with k points can have, with $k \in \{1, \ldots, n\}$? Each of these problems can be formulated as an Integer Linear Problem (ILP) as all the point coordinates are known beforehand, and the product term induced by the volume can be replaced by a sum of logarithms.[18] For each problem, the worst possible volume is obtained by finding which coordinates lead to the worst box containing k points via binary variables.

Thiémard's method does not solve all of these to optimality, but instead relies on upper and lower bounds. An upper bound and a lower bound for each of the problems is initially found greedily, then the problem returning the worst value is considered. Its *linear relaxation* is solved, where the integer variables are replaced with continuous ones, to update the bounds. It then considers the next problem giving the worst value, and continues until the final value is found or the solution is within a desired range. Should a problem be considered a second time, it will be solved exactly. Interestingly, Thiémard finds that the initial heuristic upper bounds are very good while the lower bounds are lackluster. This corresponds exactly to our experience with a different discrepancy optimization problem in Chapter 5, suggesting that, while obtaining a provably correct solution is difficult, obtaining a satisfactory approximation should be much cheaper when dealing with star discrepancies.

3.6 Genetic Approaches

Finally, we mention two very different approaches using genetic algorithms. The first is by Shah in [Sha10], in which genetic algorithms are used to compute the star discrepancy of a point set. Without going into details, the main idea behind genetic algorithms is to keep a diverse population of solutions and making it evolve via perturbations on its elements (mutations and crossovers) and selection procedures to attempt to find the best solution. In this case, an element of the population is a point q defining a box $[0, q)$. Crossover is done by taking two box corners and swapping exactly one coordinate. Mutation is done by considering an existing point and randomly modifying its coordinates. The results described by Shah suggest this method appears competitive with Thiémard's algorithm. Nevertheless, it is far from comparing to the TA-algorithm performance-wise. The relatively poor performance of generic black-box algorithms in Chapter 9 suggests that any such

18 We immediately point out that this trick cannot be used in our optimization constructions in Chapter 5 as we do not know the coordinates initially.

algorithm would need to be carefully adapted to the discrepancy problem structure, for example by only considering critical boxes as with the TA snapping procedure or by considering solvers efficient for *multi-modal* problems.

The second use of genetic algorithms was by de Rainville and Doerr in [DR13], and is an aside in this section on discrepancy computations. Their goal was very similar to ours in Chapter 6: use computational approaches to build better low-discrepancy sets. Their work focused on finding good permutations of the generalized Halton sequence. As mentioned in Section 2.3, the Halton sequence is defined by reverse b-ary decompositions, where b is prime. In the generalized sequence, rather than having $x_j^{(i)} = \sum_{k=1}^{l} d_{j,l}(i)b^{-l}$ as the reverse b-ary decompositions we have $x^{(i)} = \sum_{k=1}^{l} \pi_b(d_{j,l}(i))b^{-l}$ where π_b is a fixed permutation. Their optimization focuses on finding the optimal permutations. Without going into detail on their rather normal genetic algorithm, they obtained excellent results, providing solutions for three open problems on the inverse star discrepancy described in [NW10][Open Problem 42].[19] Once again, we point out that their discrepancy values were computed with the TA heuristic introduced in Section 3.4, and therefore are not guaranteed to be exact.

19 They used Hinrichs' lifting procedure [Hin13] for the higher dimensional results.

4 Energy Approaches

This short chapter describes some very recent results on the use of energies to construct low-discrepancy sequences in dimension 1. While especially relevant for Chapter 8, this also sets the background for recent developments that led to a more general energy formulation which we compare with the sets obtained via subset selection in Section 7.3.4. A brief introduction to the L_2 discrepancy is given as a prelude to the description of the Kritzinger sequence. This measure will be especially relevant in Section 6.7 and Chapter 8, as well as being ubiquitous in our future research in Chapter 11.

4.1 The L_2 Discrepancy

As seen in the previous chapter, a major drawback of the L_∞ star discrepancy is how difficult it is to evaluate. While not all L_p measures for $p \in [2, +\infty($ are convenient, the case $p = 2$ proves very easy to work with.

Replacing p by 2 in equation (2.2) gives us the following formula

$$d_2^*(P) := \sqrt{\int_{[0,1)^d} \left(\frac{|P_k \cap [0,q)|}{k} - \lambda(q) \right)^2 dq}.$$

Explicitly writing out the left fraction and splitting the integral into three, Warnock gives an explicit formula that depends only on the points' coordinates and the size of P in [War72]

$$d_2^*(P) := \frac{1}{3^d} - \frac{2^{1-d}}{n} \sum_{i=1}^{n} \prod_{k=1}^{d} (1 - (x_k^{(i)})^2) + \sum_{i,j=1}^{n} \prod_{k=1}^{d} (1 - \max(x_k^{(i)}, x_k^{(j)})). \tag{4.1}$$

From a computational perspective, this is much easier to work with than the L_∞ star discrepancy as it requires only $O(dn^2)$ time to compute. A better algorithm for low dimensions was suggested in [Hei96], that requires only $O(n \log(n)^d)$ operations. This is generally better, except for high d and low n. Ease of computation

has led to the L_2 star discrepancy's prevalence, its main role being a surrogate of the L_∞ star discrepancy.[20]

The L_2 discrepancy does have a major known drawback in that, for low n in higher dimensions, it is practically optimal to place all the points in $(1, \dots, 1)$. This was shown by Matoušek in [Mat98]. Nevertheless, this should not be seen as a blocking point towards its use, and personal discussion with N. Kirk suggests that using a smoothed L_2 discrepancy optimization via machine learning can lead to excellent sets for the L_∞ star discrepancy. The Kritzinger sequence [Kri22] in Section 4.3, as well as our own work in Chapter 8, will provide further evidence of this.

4.2 One-Dimensional Greedy Constructions

As mentioned in Section 2.2, there is still a gap between the lower bound and the best constant known so far for the optimal discrepancy order for a one-dimensional sequence. More importantly, in higher dimensions even the logarithm exponent is unknown. While the $\log^d(n)/n$ conjecture would match current constructions and satisfy number theoreticians, the $\log^{d/2}(n)/n$ would correspond more to conjectures from Harmonic Analysis and Probability Theory.

In a bid to find new approaches to tackle this problem, Steinerberger proposed new *greedy* methods of constructing point sets in [Ste19; Ste20]. The goal is to dynamically construct a sequence $(x_n)_{n\in\mathbb{N}}$, by defining the newest point in the sequence x_{n+1} as

$$x_{n+1} := \operatorname*{arg\,min}_{x\in[0,1)} \sum_{1\le i\le n} f(|x - x_i|),$$

where f is an appropriately chosen function.[21] In the one-dimensional case in [Ste20], the function f is obtained via the Erdős-Turán inequality, equation (2.6), to obtain the following equation.

$$x_{n+1} := \operatorname*{arg\,min}_{x \text{ s.t. } \min_k |x-x_k|>n^{-10}} \sum_{i=1}^{n} (1 - \log(2\pi \sin(x - x_i)))$$

He shows that it is always possible to pick the next point such that the resulting

20 This should not be seen too reductively. There is a very large demand for such a surrogate and, like many other discrepancies, it does appear in certain Koksma-Hlawka type inequalities for specific classes of functions [DP14, Chapter 9].

21 Finding the function f that minimizes the discrepancy of the resulting sequence is in itself a fascinating question which will not be tackled in this thesis.

sequence has discrepancy at most $O(\log(n)/\sqrt{n})$. More importantly, numerical experiments suggest it performs comparably to, if not better than, traditional low-discrepancy sequences. This leads to his conjecture that the sequence has discrepancy of order $O(\log(n)/n)$, the traditional order of low-discrepancy sequences in one dimension. Furthermore he noticed two very interesting properties. The first is that the sequence obtained seems to be robust relative to the starting set. Even beginning with a few points very close together leads to a good sequence after a few hundred points. This will be further discussed in Chapter 8. The second is that with specific starting points, the greedy energy approach gives exactly elements of the van der Corput sequence.

It was shown very quickly afterwards by Pausinger [Pau21] that this last property is not limited to Steinerberger's functional, but is obtained as long as the functional f verifies three properties:

- f is symmetric, $f(x) = f(1-x)$,
- f is twice differentiable,
- $f''(x) > 0$ for all $x \in [0, 1)$.

Should f verify these three properties, starting the sequence with $x_0 = 0$ leads to the sequence being a *permutation* of the van der Corput sequence. With an added condition, Pausinger obtains a $O(n^{-1/3})$ bound on the discrepancy of the resulting sequence. While this bound is relatively weak, it is interesting that we are able to obtain a bound for a very general class of functionals when our bounds on specific functionals are still far from the $O(\log(n)/n)$ target. In any case, the connection to the van der Corput sequence, which is well understood and has a discrepancy in $O(\log(n)/n)$, brings better guarantees on the discrepancy of the generated sequence.

This should not be seen as a purely one-dimensional focus. Steinerberger's greedy construction can be adapted to obtain a functional that *shifts* point sets in any dimension [Ste19]. As in the one-dimensional case, this method both provides seemingly low-discrepancy sets and allows the correction of badly distributed sets.

4.3 The Kritzinger Sequence

Building on these papers, Kritzinger introduced a new functional in [Kri22], based on the L_2 discrepancy. Unlike for the L_∞ star discrepancy, the contribution $F(y, P)$ of

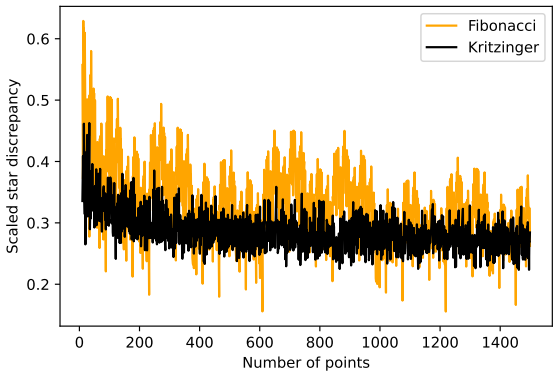

Figure 4.1: The Kritzinger sequence compared to the Fibonacci sequence over the first 1500 points. x-axis is the number of points, while the y-axis represents the L_∞ star discrepancy of the set, scaled by $n/\log(n)$. The Fibonacci sequence is one of the most regular one-dimensional sequences, and yet the Kritzinger sequence is comparable over the whole range of values tested.

a new point y added to P is easy to express using Warnock's formula (equation (4.1)),

$$F(y,P) := -2^{1-d}\prod_{k=1}^{d}(1-y_k^2) + \frac{1}{n+1}\prod_{k=1}^{d}(1-y_k) + \frac{2}{n+1}\sum_{j=1}^{n}\prod_{k=1}^{d}(1-\max(y_k, x_k^{(j)})). \tag{4.2}$$

Kritzinger thus defines his sequence by choosing as the next point the point that minimizes the L_2 discrepancy of the new set,

$$x_{n+1} := \arg\min_{y\in[0,1)^d} F(y,P).$$

This sequence can be defined in any dimension, and calculating the contribution of a single point can be done in $O(nd)$ time. Kritzinger limited his study to dimension 1 and showed promising empirical performance of the sequence. Importantly, he showed that the next point generated, say the $(n+1)$-th, could only take specific values, $(2i+1)/2(n+1)$ where $i \in \{0,\dots,n-1\}$. This can be easily shown by noticing that the one-dimensional version of $F(y,P)$ is a degree-two polynomial with positive second order coefficient. Kritzinger showed that the sequence had a L_∞ star discrepancy of order at most $O(n^{-1/2})$ in any dimension. This is close to the bounds for random points described in Section 2.2.

Numerical experiments by Kritzinger and Steinerberger show that the sequence

is comparable to the Fibonacci sequence at least up to roughly 1500 points, as can be seen in Figure 4.1. This suggests that, once again, we do not have the appropriate bounds and Kritzinger conjectures that the correct order is $O(\log(n)/n)$. The only improvement to Kritzinger's result was by Steinerberger, who showed in [Ste24] that a quantity slightly weaker than the discrepancy is at most $O(n^{-2/3})$.[22] This was done by showing that greedy Wasserstein W_2-distance minimization leads a Kritzinger sequence, and this sequence has the stated bound for the L_∞ star discrepancy for infinitely many n. It is somewhat surprising that greedy L_2 minimization would lead to such good results for the L_∞ star discrepancy, especially in light of the weakness mentioned in the previous section and the fact that greedy L_∞ minimization does not perform well. Our recent research on the Kritzinger sequence will be presented in Chapter 8.

22 More precisely, there are infinitely many n such that the discrepancy is of order $O(n^{-2/3})$.

Part II

Contributions

5 Optimal Set Construction

This chapter is based on a mix between the long version presented in the preprint [Clé+23a], ***Constructing Optimal L_∞ Star Discrepancy Sets*** *and a more concise paper under submision at the time of writing. This was joint work with Carola Doerr, Kathrin Klamroth and Luís Paquete.*

5.1 Summary of Results

We present in this chapter two different non-linear programming formulations to obtain optimal point sets with respect to the L_∞ star discrepancy. Section 5.2 introduces the key results in dimension 2, that is both formulations and some key lemmas that greatly help the solver. In particular, we generalize a lower bound by White in [Whi77] on the L_∞ star discrepancy of small point sets in dimension 2. The obtained point sets are far better than Fibonacci sets, one of the best two-dimensional sets, with a near 50% improvement in all the cases in 2 dimensions. Table 5.1 and Figure 5.1 highlight the clear improvement in discrepancy values brought by our optimal sets. The point set structure is also very different, as shown earlier by Figure 1.1, suggesting a possible new point set construction approach.

The framework introduced for the L_∞ star discrepancy in dimension 2 can be extended to many different settings. We provide results for dimension 3 and for different measures such as the extreme and periodic discrepancies in Section 5.3. In particular, we introduce the multiple-corner discrepancy, a new discrepancy measure that aims to reduce the dependency on the corner in 0 without adding the very high computational cost of the extreme discrepancy. We show that optimizing for this measure provides sets with a more symmetric structure, while still being competitive with our optimal sets for the L_∞ star discrepancy.

More than the precise discrepancy values, the two key takeaways from this chapter should be the flexibility of the methods used and the very different structure of sets obtained.

We conclude this short summary with a technical comment. While the desired point sets should be in $[0,1)^d$, solvers such as Gurobi [Gur23] do not handle well strict inequalities. We will therefore search for optimal sets in $[0,1]^d$. Nevertheless, it is common in discrepancy theory to change to $[0,1]^d$ if more convenient and

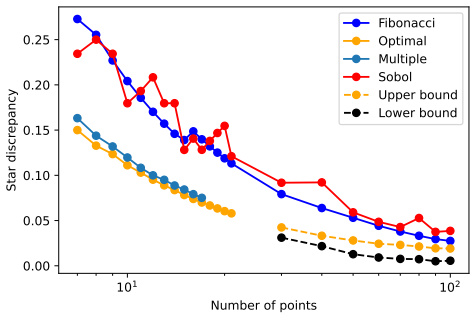

Figure 5.1: L_∞ star discrepancies for the L_∞ star optimal sets (line "optimal") and our multiple-corner optimal sets (line "multiple"), compared to the Fibonacci set. The dashed lines are lower and upper bounds described in Table 5.4.

.

depending on the context [Owe23, Section 15.1]. The lower bound we prove in Section 5.2.3 also shows that our sets will be in $[1/n, 1]$, hence avoiding the main issue that may arise with periodicity and $0 \equiv 1 \mod 1$. Nevertheless, one should be cautious with this aspect when using our models for the periodic discrepancy.

Finally, all of our code and some figures are available at https://github.com/frclement/OptiSetsDiscrepancy.

5.2 Problem Formulations in Two Dimensions

We introduce in this section our models to construct optimal L_∞ star discrepancy sets. Section 5.2.1 generalizes a result by White [Whi77] lower-bounding the star discrepancy of small sets. Section 5.2.2 presents the more intuitive model, where we are directly optimizing point positions. It is in general the better performing model in 2 dimensions. In this model, we initially make an assumption of a minimal gap

Table 5.1: Comparison of previously best values for low-discrepancy sets and our optimal sets. Optimal values were previously known for $n \leq 6$.

n	3	4	5	6	7	8	9	10	11
Optimal	0.2847	0.2500	0.2000	0.1667	0.1500	0.1328	0.1235	0.1111	0.1030
Fibonacci	0.5880	0.4910	0.3528	0.3183	0.2728	0.2553	0.2270	0.2042	0.1857
n	12	13	14	15	16	17	18	19	20
Optimal	0.0952	0.0889	0.0837	0.0782	0.0739	0.06996	0.06667	0.0634	0.0604
Fibonacci	0.1702	0.1571	0.1459	0.1390	0.1486	0.1398	0.1320	0.1251	0.1188

on the coordinates of two different points. Section 5.2.3 proves that this hypothesis does not increase the optimal L_∞ star discrepancy value by showing not only that there exist optimal point sets with distinct coordinates but also that it is possible to set the minimal coordinate in each dimension to $1/n$ when $n \geq 4$. It also describes a general procedure to shift points without increasing the discrepancy value of the set. Section 5.2.4 presents a second model, based on the fact that any point set naturally defines a grid Γ as introduced in Section 3.1. Optimizing the placement of this grid and adding constraints enforcing exactly one point per grid line/column is equivalent to optimizing the placement of the point set. Finally, we present our results for these two models in Section 5.2.5 and provide a visual comparison of our optimal sets to known low-discrepancy sets in Section 5.2.6.

5.2.1 A Generalization of a Result in [Whi77]

As mentioned in Section 2.2.2, White [Whi77] gave optimal discrepancy values for $n \leq 6$ in dimension 2. His paper also includes a lower bound on the optimal L_∞ star discrepancy values in dimension 2 for $n \geq 6$ (see Proposition 1 in [Whi77]).

Given the relevance of Proposition 1 from [Whi77] for our models, we provide below a second proof of Proposition 1, that we furthermore generalize to any dimension. This result will be used to provide a lower bound constraint to the solver. It is particularly important to our models as the solver struggles to find good lower bounds initially.

▶ **Theorem 5.1.** [Whi77, Proposition 1] Let $P \subset [0,1)^d$ with $|P| = n$. If $d = 2$ and $n \geq 4$, or $d \geq 3$ and $n \geq 3$, then $d^*_\infty(P) \geq 1/n$. ◀

Proof. Let n, d, P be as required. There are two cases to consider: whether there exist x and y in P such that neither $x \leq y$ nor $y \leq x$ coordinate-wise, or not. We will show that in both cases there exists a box with discrepancy at least $1/n$.

- We first suppose that the points in P can be ordered such that $x^{(1)} \leq x^{(2)} \leq \ldots \leq x^{(n)}$. For all the large open boxes reaching an outer edge of $[0,1]^d$ to have discrepancy smaller than $1/n$, each coordinate $x^{(i)}_j$ has to be smaller than i/n. Given this, in the best possible case, the smallest closed box containing $x^{(i)}$ has volume smaller than $(i/n)^d$ and contains i points. In dimension 2, for $i = 2$ and $n \geq 4$, we then have a local discrepancy of the smallest closed box containing $x^{(2)}$ of more than $\frac{2}{n} - \frac{2^d}{n^d}$, which is at least $1/n$. The same result holds in dimension at least 3 with the same value of i and $n \geq 3$.
- If the points are not pairwise dominating each other, there exist x and y in P such that $x_1 > y_1$ and $x_2 < y_2$ (without loss of generality on the dimensions).

The box $[0, q)$, where $q_1 = x_1$ and $q_2 = y_2$ and $q_j = \max(x_j, y_j)$ for $j \in \{3, \ldots, d\}$, contains at least two fewer points than $[0, q]$ and has the same Lebesgue measure V_q. Let k be the number of points inside $[0, q)$. Then we have $d^*_\infty(P) \geq \max\{|V_q - k/n|, |V_q - (k+2)/n|\} \geq 1/n$. One of the two boxes therefore has discrepancy at least $1/n$.

We have thus shown that in both cases, there exists a box with local discrepancy at least $1/n$, which concludes the proof. ■

By a similar argument as in Case 2, one can show that any set in dimension d with d mutually non-dominating points has discrepancy at least $d/(2n)$. We conjecture this to be a lower bound for all sets, provided n is large enough while still being far smaller than the exponential number of points based on [BLV08].

5.2.2 A "Classical" Formulation

We consider the problem of locating a set P of n points $x^{(i)} := (x_{2i-1}, x_{2i}) \in [0, 1]^2$ for $i = 1, \ldots, n$, such that[23]

$$P = \operatorname*{arg\,min}_{X=\{x^{(1)},\ldots,x^{(n)}\}} d^*_\infty(X).$$

As mentioned in the previous summary, we make the assumption that no two points in P have the same value in any coordinate. The core idea behind the model is to lower-bound the discrepancy value by the local discrepancies defined by critical boxes, and for this we need to know how many points are inside each box and where those boxes are.

Without loss of generality we assume that $x_{2i-1} < x_{2i+1}$ for all $i = 1, \ldots, n-1$, i.e., the points are labeled in increasing order of their first components. The point set P then induces a grid $\overline{\Gamma}(P)$ with grid points $g_{ij} = (x_{2i-1}, x_{2j})$ for $i, j = 1, \ldots, n$. By equation (3.4), to compute $d^*_\infty(P)$ for a point set P, we have to consider all critical grid points $g_{ij} = (x_{2i-1}, x_{2j})$ for which the defining points $x^{(i)}$ and $x^{(j)}$ are located on or just below the axes defining $g_{i,j}$. The first case corresponds to closed boxes, constraints (5.5a), where $i = j$ is possible. The second case corresponds to open boxes, constraints (5.5b), where the points can also be defined by the outer edges of the box $[0, 1)^d$. Furthermore, to avoid having to treat the boxes with a coordinate equal to 1 as separate cases, we can add two dummy points $x^{(0)} = (0, 1)$ and $x^{(n+1)} = (1, 0)$, from which we only need the coordinates x_0 and x_{2n+1}. These

23 We warn the reader that a variable x_i in the model corresponds to the coordinate of a point $x^{(j)}$, and not directly to a point of P.

have fixed values equal to 1. We do not include them in any of the sums counting the number of points inside a box as they do not represent a real point of the final set.

For each of these critical boxes, we need to determine which points of P are inside. While it is easy to decide whether $x^{(i)}$ is below $x^{(j)}$ in the first component by simply comparing the respective indices (recall that by assumption, $x_1^{(i)} = x_{2i-1} < x_{2j-1} = x_1^{(j)}$ if and only if $i < j$), we need to define indicator variables $y_{ij} \in \{0, 1\}$ for all $i, j \in \{1, \dots, n\}$ to indicate that $x^{(i)}$ is below, or equal to, $x^{(j)}$ in the second component (note that equality is only possible if $i = j$). In other words, we want that $y_{ij} = 1$ if and only if $x_2^{(i)} = x_{2i} \leq x_{2j} = x_2^{(j)}$. Then, a grid point g_{ij} needs to be considered whenever $j \leq i$ and $y_{ij} = 1$ (closed boxes), and whenever $j < i$ and $y_{ij} = 1$ (open boxes).

In order to properly define the indicator variables y_{ij} for all $i, j \in \{1, \dots, n\}$, we first consider the case that $i < j$. Towards this end, observe that $x_{2i} > x_{2j}$ and hence $y_{ij} = 0$ if and only if $x_{2j} - x_{2i} < 0$. This can be translated into linear constraints on the indicator variable y_{ij} as follows:

$$\begin{array}{llll} (x_{2j} - x_{2i} < 0 & \Rightarrow & y_{ij} = 0) & \text{is enforced by } x_{2j} - x_{2i} > y_{ij} - 1 \\ (x_{2j} - x_{2i} < 0 & \Leftarrow & y_{ij} = 0) & \text{is enforced by } x_{2j} - x_{2i} < y_{ij}. \end{array}$$

By anti-symmetry, we have $y_{ij} = 1 - y_{ji}$ for all $i, j = 1, \dots, n$, $i \neq j$. Moreover, we set $y_{ii} = 1$ for all $i = 1, \dots, n$. These constraints correspond to constraints (5.5d) to (5.5g) in the model below. Note that in the formulation below, we additionally require a small distance $\varepsilon > 0$ between x-coordinate (constraints (5.5c)) and y-coordinate values (constraints (5.5d)) of distinct points in order to avoid degenerate situations with different points with one or multiple equal coordinates. We show in Section 5.2.3 that there is at least one optimal solution verifying this for ε small enough. A solution to the model with this small enough ε is then a provably optimal solution of our initial problem.

We hence obtain the following nonlinear programming problem (that we refer to as model (5.5) in the following) that has quadratic terms in the constraints due to the volume computations:

$$\begin{aligned} \min \quad & f \\ \text{s.t.} \quad & \frac{1}{n} \sum_{u=1}^{i} y_{uj} - x_{2i-1} x_{2j} \leq f + (1 - y_{ij}) \qquad \forall i, j = 1, \dots, n,\ j \leq i \qquad (5.5a) \end{aligned}$$

$$\frac{-1}{n}\left(\sum_{u=0}^{i-1} y_{uj} - 1\right) + x_{2i-1}x_{2j} \leq f + (1 - y_{ij}) \qquad \forall i = 1, \ldots, n+1,\ j < i \qquad (5.5b)$$

$$x_{2i+1} - x_{2i-1} \geq \varepsilon \qquad \forall i = 1, \ldots, n-1 \qquad (5.5c)$$

$$x_{2j} - x_{2i} \geq y_{ij} - 1 + \varepsilon \qquad \forall i, j = 1, \ldots, n,\ i < j \qquad (5.5d)$$

$$x_{2j} - x_{2i} \leq y_{ij} \qquad \forall i, j = 1, \ldots, n,\ i < j \qquad (5.5e)$$

$$y_{ij} = 1 - y_{ji} \qquad \forall i, j = 1, \ldots, n,\ i > j \qquad (5.5f)$$

$$y_{ii} = 1 \qquad \forall i = 1, \ldots, n \qquad (5.5g)$$

$$x_0 = x_{2n+1} = 1$$

$$y_{0j} = 0 \qquad \forall j = 1, \ldots, n$$

$$y_{j0} = y_{(n+1),j} = 1 \qquad \forall j = 0, \ldots, n$$

$$x_i \in (0, 1] \qquad \forall i = 1, \ldots, 2n$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j = 1, \ldots, n$$

$$f \geq 0.$$

Note that the variables $y_{j,(n+1)}$ for $j = 0, \ldots, n+1$ are never used in the model and are hence not defined.

As mentioned previously, constraints (5.5a) and (5.5b) only need to be enforced for critical grid points. We thus enforce them only for $j \leq i$ for closed boxes, accounting for the case that a point defines a critical grid point by itself, and for $j < i$ in the case of open boxes. Moreover, the constraints are relaxed (by adding 1 on the right-hand side) whenever $x^{(j)}$ is not above (or equal to) $x^{(i)}$ (and hence $x^{(i)}$ is not below g_{ij} and the box is not critical), i.e., whenever $y_{ij} = 0$. For constraints (5.5b), we need to count all points in the respective volume that are strictly below the considered grid point g_{ij}. Since we assume that no two points share the same coordinate value in any dimension, the summation is over the indices 1 to $i - 1$, where we have to correct for the case that $u = j$ for which we consider the point $x^{(j)}$ with $y_{jj} = 1$ by subtracting 1 from the sum over the y_{uj}'s. Boxes with a coordinate equal to 1 correspond to the special cases where $i = n + 1$ or $j = 0$ in constraints (5.5b).

Constraints (5.5c) (for the first dimension) and (5.5d) and (5.5e) (for the second dimension) impose a minimum coordinate difference ε. Since the points are ordered in the first dimension, only consecutive pairs need to be checked there. Section 5.2.3 justifies this minimal coordinate difference.

In order to strengthen model (5.5), additional constraints may be added.

$$f \geq 1/n \qquad \text{valid if } n \geq 4 \qquad (5.5h)$$

$$y_{ij} + y_{jk} - 1 \leq y_{ik} \qquad \forall i, j, k = 1, \ldots, n \qquad (5.5i)$$

$$y_{ij} + y_{jk} \geq y_{ik} \qquad \forall i, j, k = 1, \ldots, n \qquad (5.5j)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} = \frac{n(n+1)}{2} \qquad (5.5k)$$

This includes, among others, known bounds on the optimal value of f as well as constraints that are based on the specific properties of the sorting variables y_{ij} as, e.g., transitivity. [Whi77] derived a lower bound $d^*_\infty \geq 1/n$ value for optimal points sets with $n > 6$ in dimension 2, and that we generalize to $n > 4$ in any dimension in Theorem 5.1. This knowledge can be included in the model by adding constraint (5.5h). Constraints (5.5i) and (5.5j) enforce transitivity for the indicator variables, i.e., if $x_2^{(i)} = x_{2i} < x_{2j} = x_2^{(j)}$ and $x_2^{(j)} = x_{2j} < x_{2k} = x_2^{(k)}$ (i.e., if $y_{ij} = 1$ and $y_{jk} = 1$), then also $x_2^{(i)} = x_{2i} < x_{2k} = x_2^{(k)}$ (i.e., $y_{ik} = 1$). Note that constraints (5.5i) are also valid for cases where some or all of the indices i, j, k are equal. Constraints (5.5j) cover the converse case where $x_2^{(i)} > x_2^{(j)}$ and $x_2^{(j)} > x_2^{(k)}$ (i.e., if $y_{ij} = 0$ and $y_{jk} = 0$) and hence $x_2^{(i)} > x_2^{(k)}$ (i.e., $y_{ik} = 0$). Finally, constraint (5.5k) counts the number of y_{ij} variables that have the value 1 in any feasible solution, which is always constant.

5.2.3 Minimal Point Spacing

In order to refine the model further and justify our general position hypothesis, we prove in this section some properties on optimal point sets. In particular we show that there exists some optimal sets in dimension 2 such that

- No two points share the same coordinate value in any of the coordinates (we refer to this as the *general position assumption*).
- Either the points all have a minimum distance from the lower and left boundaries ((j, i, δ)-shifts in Lemma 5.3), or the respectively first points are on the lower and left boundaries of the unit cube ($(j, i, -\delta)$-shifts in Lemma 5.5). This results holds in any dimension.
- The points satisfy some minimum distance requirements, i.e., they can be moved such that the distance between two vertically consecutive points / horizontally consecutive points is at least ε with a sufficiently small $\varepsilon > 0$.

▶ **Definition 5.2.** Consider a point set $P = \{x^{(1)}, \ldots, x^{(n)}\}$. Let $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, d\}$, and let $0 \leq \delta \leq 1 - x_j^{(i)}$. A (j, i, δ)-shift corresponds to the set

obtained by replacing the j-th coordinate of the i-th point $x_j^{(i)}$ by $x_j^{(i)} + \delta$. A (j, i, δ)-shift is called *admissible* if

$$\delta \leq \frac{1}{n} - \min_{k \neq i : x^{(k)} \leq x^{(i)}} x_j^{(i)} - x_j^{(k)}$$

if $\{k \neq i : x^{(k)} \leq x^{(i)}\}$ is not empty and $\delta \leq 1/n - x_j^{(i)}$ otherwise. ◀

Figure 5.2 provides an example of an admissible $(1, 8, 0.09447)$-shift.

▶ **Lemma 5.3.** Let P be a point set of $[0, 1)^d$ and let $s_{j,i,\delta}(P)$ be the set obtained after an admissible (j, i, δ)-shift of P. Then $d_\infty^*(P) \geq d_\infty^*(s_{j,i,\delta}(P))$. ◀

The key argument behind this result is to show that if a (j, i, δ)-shift is admissible, then no open box defined by $x^{(i)}$ was critical. Indeed, in this case there exists a smaller open box, that was dominated by the point $x^{(i)}$ that we are shifting, and that had worse discrepancy than the largest open box not containing $x^{(i)}$. The j-coordinate difference between this box and those defined by our i-th point then leads to the bounds on δ for admissibility.

Proof. Consider an admissible (j, i, δ)-shift from $P^* = \{x^{(1)}, \ldots, x^{(n)}\}$ to a new n-point set $\hat{P} := s_{j,i,\delta}(P^*)$ with $i \in \{1, \ldots, n\}$ arbitrary but fixed. Without loss of generality, we will consider $j = 1$ and the numbering of the points to correspond to the ordering in the first dimension. We refer to the grid points induced by P^* as $\overline{\Gamma}(P^*) := \{g_{ab} : a \in \{1, \ldots, n+1\}, b \in \{1, \ldots, n+1\}^{d-1}\}$ and to the grid points induced by $\hat{P}$ as $\overline{\Gamma}(\hat{P}) := \{\hat{g}_{ab} : a \in \{1, \ldots, n+1\}, b \in \{1, \ldots, n+1\}^{d-1}\}$, where the index $n+1$ is associated to value 1 in the respective coordinate. To show that discrepancy has not increased we only need to verify that it has not increased for the grid points in equation (3.4).

We first consider the case of closed boxes. For $a \in \{1, \ldots, n+1\} \setminus \{i\}$ and $b \in \{1, \ldots, n+1\}^{d-1}$, the closed box defined by grid point $\hat{g}_{ab} = g_{ab}$, contains either the same number of points or one less. The local discrepancy for these closed boxes can only be lower. For a box $\hat{g}_{ib}$, either we did not cross a hyperplane defined by $x_1 = x_1^{(k)}$ with $k > i$ and we have the same number of points in a larger volume, or there existed some $k > i$ such that $\hat{g}_{ib}$ contains as many points as g_{kb} and the volume associated with g_{kb} is smaller. In both cases, the local discrepancy associated with $\hat{g}_{ib}$ is smaller than $d_\infty^*(P^*)$.

We now consider open boxes. As before, $\hat{g}_{kb} = g_{kb}$ for $k < i$ and the points inside are unchanged. Boxes $\hat{g}_{ab}$ for $a > i$ are either unchanged or contain one less point than before the shift. If they contain one less point, then there exists $k < i$ such

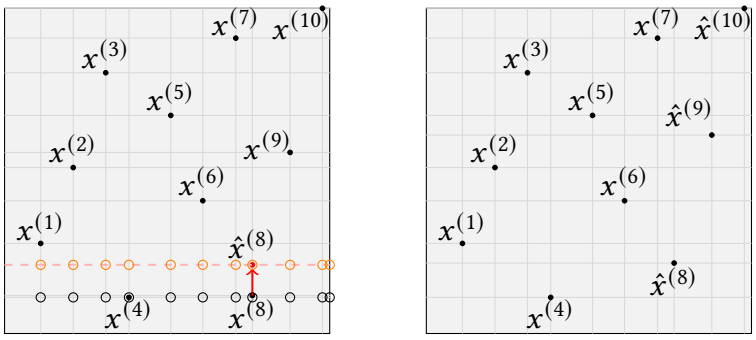

Figure 5.2: Left: An optimal 10-point set P^* with $f^* = d^*_\infty(P^*) = 0.1111$. The red arrow indicates the $(1, 8, 0.09447)$-shift, which is admissible with $\delta = \frac{1}{n} - (x_2^{(8)} - x_2^{(4)})$. Note that $x^{(8)}$ is slightly higher than $x^{(4)}$, i.e., the two points are not on the same horizontal grid line. Right: Alternative optimal 10-point set after implementing all admissible (j, i, δ)-shifts.

that $x_1^{(i)} \leq x_1^{(a)} \leq x_1^{(k)} + 1/n$ and $x^{(k)} \leq x^{(i)}$. Therefore, the box defined by $\hat{g}_{kb} = g_{kb}$ has volume less than $1/n$ smaller than $\hat{g}_{ab}$ and contains at least one less point: $x^{(k)}$. Hence, the local discrepancy for $\hat{g}_{ab}$ is smaller than $d^*_\infty(P^*)$. The last box type to consider is $\hat{g}_{ib}$. A similar argument as above can be used: this box will have volume at most $1/n$ greater than a certain g_{kb} and contains at least one more point.

We have shown that for all boxes appearing in equation (3.4) for $\hat{P}$ there exists a box with worse local discrepancy for P^*, which concludes the proof. ■

We are also able to formulate an equivalent definition for negative values of δ. The admissibility then depends on points that dominate $x^{(i)}$ rather than points dominated by $x^{(i)}$.

▶ **Definition 5.4.** Consider a point set $P = \{x^{(1)}, \ldots, x^{(n)}\}$. Let $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, d\}$, and let $0 \leq \delta \leq x_j^{(i)}$. A $(j, i, -\delta)$-shift corresponds to the set obtained by replacing the j-th coordinate of the i-th point $x_j^{(i)}$ by $x_j^{(i)} - \delta$. A $(j, i, -\delta)$-shift is called *admissible* if

$$\delta \leq \frac{1}{n} - \min_{k \neq i : x^{(k)} \geq x^{(i)}} x_j^{(k)} - x_j^{(i)}$$

if $\{k \neq i : x^{(k)} \leq x^{(i)}\}$ is not empty and $\delta \leq x_j^{(i)} - (1 - 1/n)$ otherwise. ◀

▶ **Lemma 5.5.** Let P be a point set of $[0, 1)^d$ and let $s_{j,i,-\delta}(P)$ be the set obtained after an admissible $(j, i, -\delta)$-shift of P. Then $d^*_\infty(P) \geq d^*_\infty(s_{j,i,-\delta}(P))$. ◀

The proof is analogous to the one of Lemma 5.3.

▶ **Corollary 5.6.** For $d = 2$ and $n \in \mathbb{N}_{>0}$, there exists a set in general position with discrepancy $d^*_\infty(n, 2)$. ◀

Proof. Let P be a point set in $[0, 1)^2$ with discrepancy $d^*_\infty(n, 2)$. If there is a single point in P, the result is trivial. Otherwise, without loss of generality, suppose there exist $x^{(1)}$ and $x^{(2)}$ such that $x^{(1)}_1 = x^{(2)}_1$. Then one of the two points dominates the other, suppose $x^{(1)} \leq x^{(2)}$. A $(1, 2, \delta)$-shift with $0 < \delta \leq 1/n$ is thus admissible and we can shift $x^{(2)}$ such that it does not share its first coordinate with any other point. Repeating this for all the points and the two dimensions gives the desired result. ■

This direct consequence of Lemma 5.3 does not generalize in the same way to three dimensions and above, as a shared coordinate for two points no longer guarantees that one point dominates the other. While it seems likely that for low n no two points should share a coordinate, we have no proof in higher dimensions.

Lemmas 5.3 and 5.5 imply that there always exists an optimal n-point set that satisfies additional "distance" constraints, while, of course, not all optimal n-point sets necessarily satisfy such constraints. In addition, existing n-point sets can be modified without deteriorating the objective value by applying admissible shifts in any sequence. This implies, among others, that in two dimensions the points that are closest to the lower and left boundary of the unit square can be moved up and right, respectively. Since there are no other points below them, they can actually be moved up to a distance of f from the respective boundaries, without increasing the discrepancy value f of the considered configuration. Moreover, for every pair of points $x^{(i)} \neq x^{(j)}$ with $x^{(i)}_1 \leq x^{(j)}_1$ and $x^{(i)}_2 \leq x^{(j)}_2$, we can apply $(2, j, \delta)$ or $(2, j, -\delta)$ admissible shifts so that the vertical distance between the two points is at least $\frac{1}{n}$, and similarly we can apply $(1, j, \delta)$ or $(1, j, -\delta)$ so that the horizontal distance between the two points is at least $\frac{1}{n}$.[24] Note that this is also possible when one (or both) of the points are close to the upper or right boundary of the unit cube, since we can apply $(1, i, -\delta)$ or $(2, i, -\delta)$ admissible shifts in this case.

The constraints (5.5l)-(5.5o) implement these conditions, constraints (5.5p) and (5.5q) follow from the inequalities for the grid points on the upper boundary, and constraint (5.5r) follows by the triangle inequality.

$$x_1 = f \tag{5.5l}$$

$$x_{2i} \geq f \qquad \forall i = 1, \ldots, n \tag{5.5m}$$

24 This **does not** mean the coordinates will be multiples of $1/n$ as we require two points where one dominates the other for this comment to hold.

$$x_{2i+1} - x_{2i-1} \geq y_{i,(i+1)} - 1 + \frac{1}{n} \qquad \forall i = 1, \ldots, n-1 \qquad (5.5n)$$

$$x_{2j} - x_{2i} \geq y_{ij} - 1 + \frac{1}{n} \qquad \forall i, j = 1, \ldots, n,\ i < j \qquad (5.5o)$$

$$x_{2i-1} \leq f + \frac{i-1}{n} \qquad \forall i = 1, \ldots, n \qquad (5.5p)$$

$$x_{2i-1} \geq \frac{i}{n} - f \qquad \forall i = 1, \ldots, n \qquad (5.5q)$$

$$x_{2j-1} - x_{2i-1} + x_{2j} - x_{2i} \geq y_{ij} - 1 + \frac{2}{n} \qquad \forall i, j = 1, \ldots, n-1,\ i < j \qquad (5.5r)$$

Note that while constraints (5.5n) can be introduced in addition to constraints (5.5c) but can not replace them, constraints (5.5o) strengthen constraints (5.5d) and (5.5e) and should thus be used instead. Finally, we note that constraints (5.5p) make some of the constraints of type (5.5b) redundant. Indeed, when $j = 0$, constraint (5.5b) gives us exactly constraint (5.5p). With this improvement, we can now set $y_{i0} = 0$ for all $i \in \{1, \ldots, n+1\}$.

5.2.4 An Assignment Formulation

We now introduce a second formulation, which will also exploit the results from the previous section. An alternative formulation is obtained when defining the two coordinates of all points in P separately and independently from each other in two vectors $x, y \in [0,1]^n$. The point set P is then obtained by assigning exactly one y-coordinate to every x-coordinate[25] and vice versa, using assignment variables $a_{ij} \in \{0,1\}$, $i, j = 1, \ldots, n$. In other words, rather than optimizing a point set, we optimize the associated grid (and indirectly the set): a point (x_i, y_j) is in P if and only if $a_{ij} = 1$. The advantage of this formulation is its simplicity, since the sorting can be implemented already when defining the x- and y-coordinates of the vertical and horizontal grid lines, respectively, in an increasing order.

$$\begin{aligned} \min\ & f \\ \text{s.t.}\ & \frac{1}{n} \sum_{u=1}^{i} \sum_{v=1}^{j} a_{uv} - x_i y_j \leq f && \forall i, j = 1, \ldots, n && (5.6a) \\ & \frac{-1}{n} \sum_{u=1}^{i-1} \sum_{v=1}^{j-1} a_{uv} + x_i y_j \leq f && \forall i, j = 1, \ldots, n+1 && (5.6b) \end{aligned}$$

25 x represents here the sorted list of the first coordinates of the points in P.

$$x_{n+1} = 1,\ y_{n+1} = 1 \tag{5.6c}$$
$$x_{i+1} - x_i \geq \varepsilon \qquad \forall i = 1, \ldots, n-1 \tag{5.6d}$$
$$y_{i+1} - y_i \geq \varepsilon \qquad \forall i = 1, \ldots, n-1 \tag{5.6e}$$
$$\sum_{i=1}^{n} a_{ij} = 1 \qquad \forall j = 1, \ldots, n \tag{5.6f}$$
$$\sum_{j=1}^{n} a_{ij} = 1 \qquad \forall i = 1, \ldots, n \tag{5.6g}$$
$$x_i, y_i \in [0, 1] \qquad \forall i = 1, \ldots, n$$
$$a_{ij} \in \{0, 1\} \qquad \forall i, j = 1, \ldots, n$$
$$f \geq 0.$$

Constraints (5.6a) and (5.6b) correspond to the discrepancy inequalities, with the double sum counting the number of selected points inside the box defined by (x_i, y_j). Constraints (5.6d) and (5.6e) impose the minimal distance between two grid lines, as derived in the previous subsection. Finally, constraints (5.6f) and (5.6g) impose that there is exactly one point in each column and row of the grid. The "exactly" comes from the general position assumption we derived previously.

Note that the values of x_{n+1}, y_{n+1} are fixed to one. These parameters are used as dummy values to include the open box constraints (5.6b) for the grid points on the upper and right boundaries in a simple way. Note also that many of the constraints (5.6a) and (5.6b) are redundant since the defining points may not be located below or on the respective grid lines. It is possible to check if a box defined by x_i and y_j is critical by calculating $\sum_{k=1}^{i} a_{kj} + \sum_{k=1}^{j} a_{ik}$. This sum is equal to 2 if and only if there is a point on each of the outer edges and thus if and only if the box is critical. In practice, the model was slower when adding this requirement. It may be worthwhile to use this requirement for other solvers or different settings.

As in the previous model, the formulation can be strengthened by the following constraints:

$$f \geq 1/n \qquad \text{if } n \geq 4 \tag{5.6h}$$
$$x_1 = f \tag{5.6i}$$
$$y_1 = f \tag{5.6j}$$
$$x_j - x_i \geq \frac{1}{n} - \left(1 - \sum_{u=1}^{k} (a_{iu} - a_{ju})\right) \qquad i, j = 1, \ldots, n,\ i < j,\ k = 1, \ldots, n \tag{5.6k}$$

$$y_j - y_i \geq \frac{1}{n} - \left(1 - \sum_{u=1}^{k}(a_{iu} - a_{ju})\right) \qquad i, j = 1, \ldots, n,\ i < j,\ k = 1, \ldots, n \tag{5.6l}$$

$$x_i \leq f + \frac{i-1}{n} \qquad i = 2, \ldots, n \tag{5.6m}$$

$$x_i \geq \frac{i}{n} - f \qquad i = 2, \ldots, n \tag{5.6n}$$

$$y_i \leq f + \frac{i-1}{n} \qquad i = 2, \ldots, n \tag{5.6o}$$

$$y_i \geq \frac{i}{n} - f \qquad i = 2, \ldots, n \tag{5.6p}$$

Constraint (5.6h) uses Theorem 5.1. Constraints (5.6i) to (5.6p) are direct analogies to constraints (5.5m) to (5.5q). Only constraints (5.6k) and (5.6l) have to be adapted since we no longer use ordering variables (the ordering is naturally defined by the grid itself).

5.2.5 Experimental Results

We describe in this section the results obtained by our different models. All experiments were run with Gurobi 10.0.0 with an accuracy of 10^{-4} using Julia and the JuMP package. Experiments were run on a single machine of the MeSU cluster at Sorbonne Université, Intel Xeon CPU E5-2670 v3 with 24 cores.

Tables 5.2 and 5.3 give the optimal discrepancy values for point sets for $n = 1$ to $n = 21$ for different models, as well as the associated runtimes. M5 corresponds to the continuous formulation described in (5.5) and M6 corresponds to the assignment formulation (5.6). The "+" then indicates which extra constraints were added. The runtime is indicated for each formulation, the returned value is logically always the same. We note that values for $n = 1$ to $n = 6$ correspond to those known previously. The continuous formulation M5 is significantly faster than the assignment one M6, both with and without the extra constraints. We note that the effectiveness of the extra constraints is debatable: initial tests on a small laptop provided a factor 4 speedup, but there is no obvious improvement for the tests on the cluster shown below for M5. We compare the discrepancies of our optimal sets to one of the most famous sets, obtained from the Kronecker sequence with golden ratio which we call Fibonacci set (see Section 2.3 for a brief description).

During the computational experiments we observed that very good – if not optimal – solutions were often found quite early during the solution process, and a large amount of time was then used by the solver to close the duality gap, i.e., to

Table 5.2: $d = 2$, $n \geq 2$ points located in the continuous box $[0, 1]^2$. All problems solved with Gurobi to global optimality with a tolerance of 10^{-4}.

n	2	3	4	5	6	7	8	9
M5	0.01	0.02	0.63	0.41	0.78	0.81	0.3	0.93
$M5_{+h\ldots r}$	0.01	0.1	0.01	0.14	0.07	0.2	0.32	0.7
M6	0.04	0.06	0.33	0.27	0.22	2.04	2.11	91.32
$M6_{+h\ldots p}$	0.05	0.05	0.38	0.29	0.33	0.58	0.80	7.19
$d^*_\infty(P_{\text{cont}})$	0.3660	0.2847	0.2500	0.2000	0.1667	0.1500	0.1328	0.1235
$d^*_\infty(Fib)$	0.6909	0.5880	0.4910	0.3528	0.3183	0.2728	0.2553	0.2270

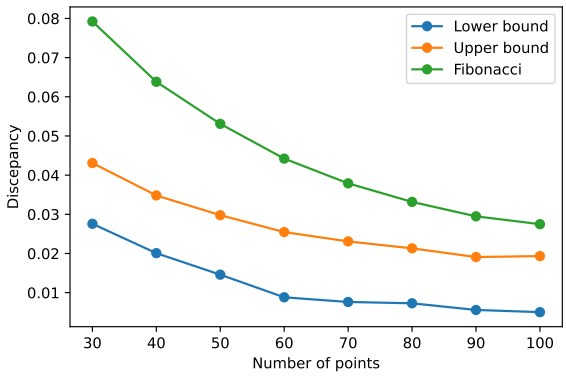

Figure 5.3: Comparison of the bounds at 40 000s obtained in Table 5.4 with the values of the Fibonacci set.

prove global optimality of the solution. Thus, very good point sets can be obtained by running a solver with a pre-specified time limit, however without showing optimality. Table 5.4 and Figure 5.3 show discrepancy values obtained for Model $M5_{+h}$ within a time limit of 1 000s, 10 000s, 20 000s and 40 000s, and a comparison with the Fibonacci sequence. It turns out that for n larger than 100, the complexity of the model increases too much to make this a viable approach. For the 100 point set, all the results before the 40 000 seconds cutoff are worse than the Fibonacci set, only the last improvement makes it noticeably better. We note that Model $M5_{+h}$ outperformed Model $M5_{+h,\ldots,r}$ in nearly all cases: only some duality gaps are smaller with 1 000 seconds runtime.

Table 5.3: $d = 2$, $n \geq 10$ points located in the continuous box $[0, 1]^2$. While in almost all instances an optimal solution was obtained very quickly, it often took a very long time to close the duality gap and to prove optimality. All problems solved with Gurobi 10.0.0 to global optimality with a tolerance of 10^{-4}.

n	10	11	12	13	14	15
M5	1.46	5.06	7.29	16.98	62.22	70
$M5_{+h\ldots r}$	0.91	4.89	12.01	21.53	69.94	110.56
$M6_{+h\ldots p}$	5.47	15.41	27.61	65.31	6 279.16	3 445.26
$d^*_\infty(P_{\text{cont}})$	0.1111	0.1030	0.0952	0.0889	0.0837	0.0782
$d^*_\infty(Fib)$	0.2042	0.1857	0.1702	0.1571	0.1459	0.1390
n	16	17	18	19	20	21
$M5_{+h\ldots r}$	426.01	616.45	4 610.23	11 240.5	21 892.76	77 988.0
$M6_{+h\ldots p}$	12 974.11	22 020.44				
$d^*_\infty(P_{\text{cont}})$	0.0739	0.06996	0.0667	0.0634	0.0604	0.0580
$d^*_\infty(Fib)$	0.1486	0.1398	0.1320	0.1251	0.1188	0.1132

5.2.6 Structural Differences between Known and Optimal Point Sets

Finally, plotting the local discrepancy values over $[0, 1]^2$ reveals that our optimal sets have a very different structure to known low-discrepancy sets. In each plot, we calculate the local discrepancy values over $[0, 1)^2$, with brighter colors indicating a worse discrepancy. Note that each plot has its own color scale. Furthermore, "triangles" whose corner is to the top-right correspond to open boxes with too few points whereas those with a corner to the lower-left correspond to closed boxes with too many points. To facilitate a visual inspection, we also provide a *truncated* version of these plots, where all local discrepancy values smaller than $d^*_\infty(P) - 1/n$ are set to 0. This allows us to see better where the worst discrepancy values are reached and if the local discrepancy values are balanced over the whole space. While in sets like Fibonacci or the initial segments of the Sobol' sequence only a few closed boxes give active constraints for the discrepancy (i.e. the inequality in the constraint is an equality), a much bigger set of boxes are very close to the maximal discrepancy value for our sets. In particular, the truncated plots show that a very large part of $[0, 1)^2$ has a local discrepancy close to $1/n$ for our optimal sets. For both Fibonacci and Sobol' sets, only overfilled boxes appear, and this seems to be a characteristic independent of n. However, for our sets, both types of triangles appear and quite often sharing a box corner.

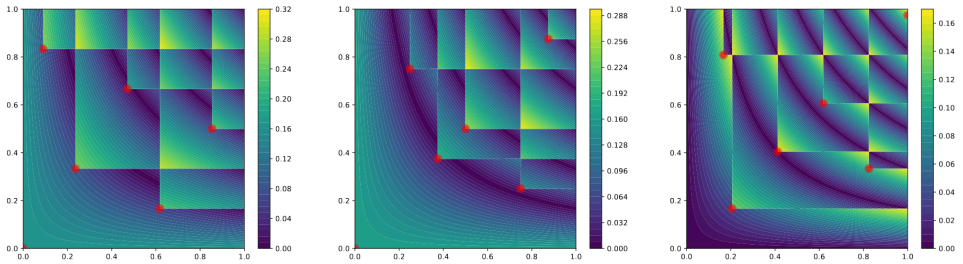

Figure 5.4: Fibonacci, Sobol' and optimal sets' local discrepancies for $n = 6$.

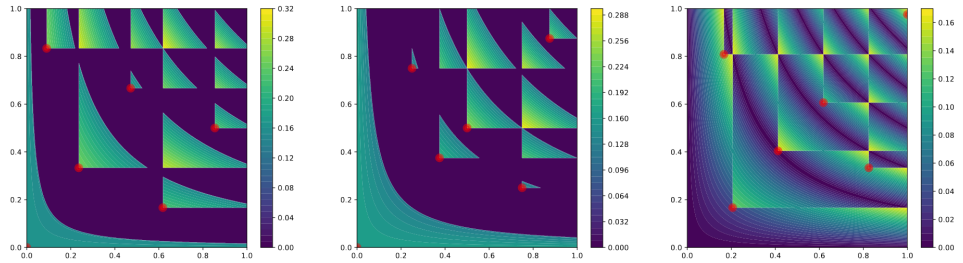

Figure 5.5: Fibonacci, Sobol' and optimal sets' *truncated* local discrepancies for $n = 6$. Local discrepancy values more than $1/n$ away from the star discrepancy were set to 0. Colored regions are therefore close to the worst discrepancy value.

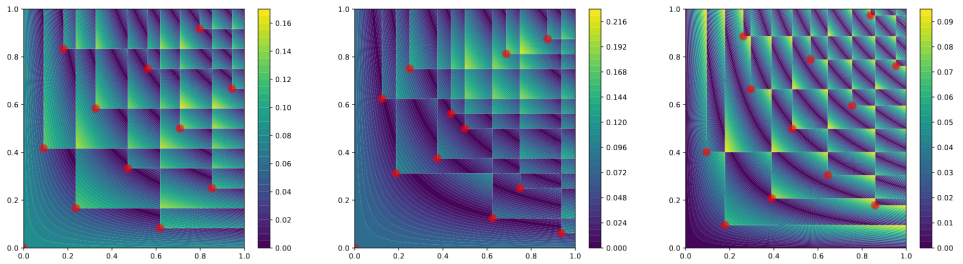

Figure 5.6: Fibonacci, Sobol' and optimal sets' local discrepancies for $n = 12$.

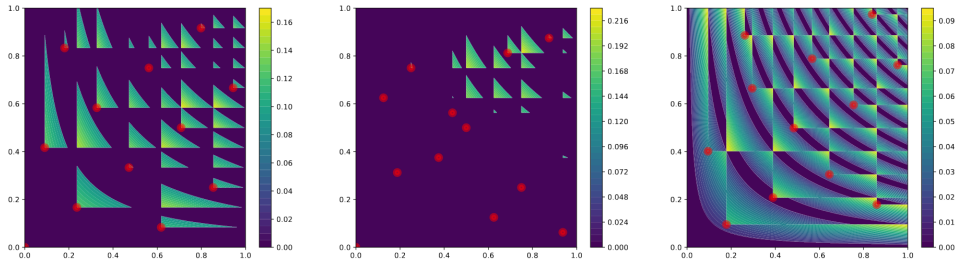

Figure 5.7: Fibonacci, Sobol' and optimal sets' truncated local discrepancies for $n = 12$.

Table 5.4: $d = 2$, points located in the continuous box $[0,1]^2$. All problems solved for Model $M5_{+h}$ with Gurobi with a time limit of 1 000s, 10 000s, 20 000 and 40 000 seconds, respectively. Gap represents the difference between the current best solution and a lower bound found by the solver.

n	30	40	50	60	80	100
time limit 1 000s:						
$d^*_\infty(P_{\text{cont}})$	0.04305	0.035	0.0317	0.02934	0.02467	0.04070
gap	0.01718	0.01780	0.02432	0.02567	0.01807	0.03568
time limit 10 000s:						
$d^*_\infty(P_{\text{cont}})$	0.0426	0.0334	0.0283	0.02469	0.02467	0.02570
gap	0.01296	0.01087	0.01550	0.01682	0.01781	0.02068
time limit 20 000s:						
$d^*_\infty(P_{\text{cont}})$	0.0426	0.0333	0.0283	0.0246	0.02379	0.02346
gap	0.01208	0.01015	0.01530	0.01682	0.01653	0.01844
time limit 40 000s:						
$d^*_\infty(P_{\text{cont}})$	0.0424	0.0332	0.028	0.02435	0.02131	0.01933
gap	0.01124	0.00947	0.01519	0.01666	0.01403	0.01431
Reference						
Fibonacci	0.079231	0.063836	0.053068	0.044223	0.033167	0.027485

5.3 Extensions

This section presents a number of possible extensions of our model. Section 5.3.1 introduces the most natural one, an extension to higher dimensions. Section 5.3.2 describes a way of obtaining optimal lattices, a very frequent structure in the discrepancy community. Finally, Section 5.3.3 considers alternative discrepancy notions (extreme, periodic, and multiple-corner discrepancy). These different measures appear frequently in the discrepancy literature [Mat10], sometimes under the name "axis-parallel" boxes for the extreme discrepancy. For example, the paper by Hinrichs and Oettershagen [HO14] mentioned in Section 2.2.2 tackled the optimal constructions for the periodic L_2 discrepancy (not star).

5.3.1 An Extension to Three Dimensions

Both models presented in Section 5.2 naturally extend to higher dimensions. However, a considerably simpler model is obtained when generalizing model (5.6) to the three-dimensional case and will be the only one described here. Our experi-

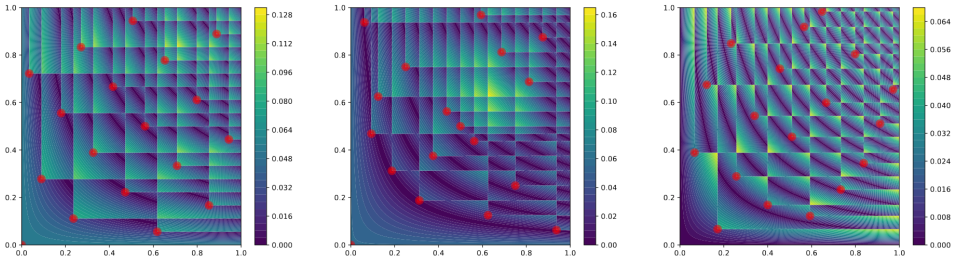

Figure 5.8: Fibonacci, Sobol' and optimal sets' local discrepancies for $n = 18$.

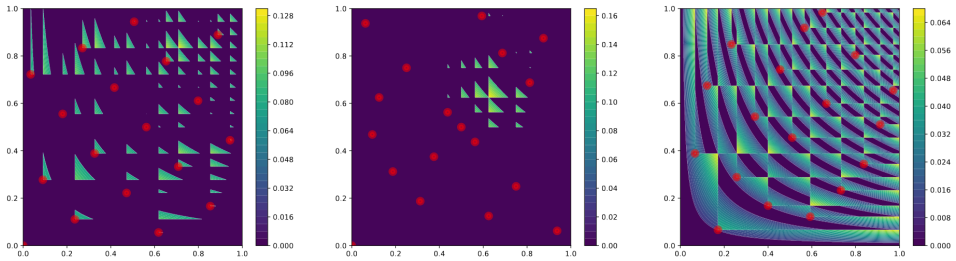

Figure 5.9: Fibonacci, Sobol' and optimal sets' truncated local discrepancies for $n = 18$.

ments suggest it is also the fastest model to solve. As in model (5.6), we define the x-, y- and z-coordinates separately and independently from each other in vectors $x, y, z \in [0,1]^n$. The point set is then obtained by assigning exactly one y- and one z-coordinate to every x-coordinate, and vice versa, using assignment variables $a_{ijk} \in \{0,1\}$, $i, j, k = 1, \ldots, n$.

$$\begin{aligned}
\min \quad & f \\
\text{s.t.} \quad & \frac{1}{n} \sum_{u=1}^{i} \sum_{v=1}^{j} \sum_{w=1}^{k} a_{uvw} - x_i y_j z_k \leq f && \forall i, j, k = 1, \ldots, n && \text{(5.7a)} \\
& \frac{-1}{n} \sum_{u=1}^{i-1} \sum_{v=1}^{j-1} \sum_{w=1}^{k-1} a_{uvw} + x_i y_j z_k \leq f && \forall i, j, k = 1, \ldots, n+1 && \text{(5.7b)} \\
& x_{n+1} = 1, \; y_{n+1} = 1, \; z_{n+1} = 1 && && \text{(5.7c)} \\
& \sum_{j=1}^{n} \sum_{k=1}^{n} a_{ijk} = 1 && \forall i = 1, \ldots, n && \text{(5.7d)} \\
& \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ijk} = 1 && \forall j = 1, \ldots, n && \text{(5.7e)}
\end{aligned}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ijk} = 1 \qquad \forall k = 1,\dots,n \qquad (5.7f)$$

$$x_i, y_i, z_i \in [0,1] \qquad \forall i = 1,\dots,n;$$

$$a_{ijk} \in \{0,1\} \qquad \forall i,j,k = 1,\dots,n$$

$$f \geq 0$$

The values of $x_{n+1}, y_{n+1}, z_{n+1}$ are fixed to one and used as dummy values.

One should be careful that our result on the general position of some optimal sets no longer holds in higher dimensions. While the admissible shifts still do not increase the discrepancy of the set (Lemmas 5.3 and 5.5 still hold), a pair of points $x^{(1)}$ and $x^{(2)}$ sharing a coordinate no longer guarantees that one dominates the other. This means we can no longer use the constraints lower-bounding the gaps between different coordinates. Furthermore, some of the local discrepancy constraints will count the points incorrectly.[26] Nevertheless, for every distinct box, there will be a constraint correctly computing the local discrepancy. Indeed, suppose there exist $x_k = x_i$ and $i < k$, two coordinates in the same dimension of different points. For the closed box defined by x_i, we will not count x_k, which leads to a lower discrepancy lower bound. However, the closed box for x_k counts these points correctly: the lower bound is unchanged. A similar argument holds by swapping the roles for open boxes.

As before, the formulation can be strengthened by the following constraints, as the lowest point in each dimension can still be shifted:

$$x_1 = f \qquad (5.7j)$$

$$y_1 = f \qquad (5.7k)$$

$$z_1 = f \qquad (5.7l)$$

$$f \geq 1/n \qquad \text{valid if } n \geq 3 \qquad (5.7m)$$

This is based on the fact that Lemmas 5.3 and 5.5 immediately generalize to three and more dimensions.

We note that extensions to higher dimensions could be obtained in a similar way. However, we would obtain $O(n^d)$ constraints just to bound the discrepancy, without even considering the products of many variables: each local discrepancy constraint has a product of d variables.

26 This always underestimates the discrepancy. An overestimation would prevent the model from working.

Table 5.5 describes our results in dimension 3. We are able to solve the model in reasonable time for $n \leq 8$. While the model is relatively easy to solve with the general position hypothesis (518.98s for $n = 8$), it is vastly more expensive without. Indeed, it took 1 569 728s to solve the problem in the $n = 8$ case. Finding an optimal solution is relatively inexpensive (around a day), but closing the optimality gap takes a lot longer.

Table 5.4 in the $2d$ case suggests this is not an insurmountable obstacle if the objective is only to obtain excellent sets, and not optimal ones. Runtime values are not included as the computer was not running exclusively this experiment and the runtime values may not be representative.

Table 5.5: $d = 3$, $n \geq 1$ points located in the continuous box $[0, 1]^3$. All problems solved with Gurobi 10.0.0, using a simple reformulation of the cubic terms into quadratic terms.

n	1	2	3	4	5	6	7	8
$d^*_\infty(P_{\text{cont}})$	0.6823	0.4239	0.3445	0.3038	0.2618	0.2326	0.2090	0.1875

5.3.2 Optimal Lattice Construction

A very brief introduction to lattices for discrepancy was given in Section 2.3, highlighting in particular the two-dimensional Fibonacci set. In general, construction methods for these lattices usually involve either component by component search, exhaustive search or *Korobov* constructions, where the lattice parameter is given by $(1, a, a^2, \ldots, a^{d-1})$ for some well-chosen $a \in [0, 1]$. Our models suggest another construction method for these lattices.[27]

We now want to build a point set $P_r = \{(i/n, \{ir\}) | i \in \{0, \ldots, n-1\}\}$, with $r \in [0, 1)$. While lattice parameters are often found by considering $r = p/n$ with p coprime with n, we can obtain even better sets by considering all reals in $[0, 1]$. This can be easily implemented in the first continuous model, by adding constraints on the relations between two consecutive points in the set. We have that $x_{2i-1} = (i-1)/n$, $x_2 = 0$ and $x_{2i} + r = x_{2(i+1)} \mod 1$ for $i \in \{1, \ldots, n-1\}$. The first two equalities pose no problem and can be directly added as is in the model. For the third, we add binary variables $(k_i)_{i \in \{2,\ldots,n\}}$, where $k_i = 1$ if $x_{2i} + r > 1$. We then obtain the equality $x_{2i} + r = x_{2(i+1)} + k_i$, which can be added to our model.

27 It would be more precise to characterize what we are constructing as Kronecker constructions, rather than the more general lattices. Indeed, we are finding optimal parameters such that the two-dimensional set obtained from a Kronecker sequence is the best possible. We will nevertheless stick with the term "lattice" for readability.

While this was implemented only for model (5.5), this change can also be done for model (5.6). For model (5.6), the only choices are which a_{1j} we pick and the associated y_j, as they will define all the other variables. equation (5.5c) can be removed since we know that the distance between two consecutive x-coordinates is $1/n$. The following constraints then need to be added to model (5.5).

$$x_{2i-1} = (i-1)/n \qquad i = 1, \dots, n \tag{5.8a}$$
$$x_2 = 0 \tag{5.8b}$$
$$x_{2i} + r = x_{2i+2} + k_i \qquad i = 1, \dots, n-1 \tag{5.8c}$$
$$k_i \geq x_{2i} + r - 1 + \varepsilon \qquad i = 1, \dots, n-1 \tag{5.8d}$$
$$k_i \in \{0, 1\} \qquad i = 1, \dots, n-1$$
$$r \in [0, 1]$$

We note that a very small ε needs to be added in constraint (5.8d) because the strict inequality is not handled well by the solver. The other possible solution is to consider the inequality without the ε and verify that there is no point with 1 as a coordinate.

Table 5.6 gives the best discrepancy values obtainable for lattice sets with one parameter fixed to $1/n$. While these are worse than the optimal values for $n = 20$, they are still better than the Fibonacci sequence by a decent margin.

Table 5.6: Best discrepancies for lattice constructions in 2d for different n. r corresponds to the lattice parameter.

n	r	$d^*_\infty(LAT_1)$	$d^*_\infty(Fib)$	Runtime (s)
20	0.653	0.094898	0.1188	7.05
25	0.64269	0.077410	0.095078	21.82
30	0.733	0.06492	0.079231	149.85
35	0.82759	0.056137	0.067913	797.66
40	0.79404	0.049594	0.063836	2349.33

This approach was then generalized to have lattices with two parameters: rather than $(1/n, \{ir\})$, we now consider $(\{ir_1\}, \{ir_2\})$. These lattices will naturally be better than the single parameter ones. The constraints added to remove the fractional part now need to be used on the first variable as well. This requires removing equation (5.8a) and adding the following equations.

$$x_1 = 0, x_2 = 0 \tag{5.8e}$$
$$x_{2i-1} + r_2 = x_{2i+1} + h_i \qquad i = 1, \dots, n-1 \tag{5.8f}$$
$$h_i > x_{2i-1} + r_2 - 1 \qquad i = 1, \dots, n-1 \tag{5.8g}$$
$$h_i \in \{0, 1\} \qquad i = 1, \dots, n-1$$
$$r_2 \in [0, 1]$$

Table 5.7 gives the two parameters as well as the best discrepancies obtained for varying n by lattices. Interestingly, the first parameter r_1 is always very close to $1/n$, while still improving noticeably on the discrepancy values from Table 5.6. It can also be seen that while the first parameters are monotonically decreasing and quite close to $1/n$, the second parameters seem to have similar values (a lot of them are around 0.725). We notice that the discrepancy of the best lattice with 37 points remains far from that of our optimal point set with 20 points. This suggests that construction methods other than lattices should be considered when trying to build high-quality low-discrepancy sets.

Table 5.7: Best discrepancies obtained for double lattices $d^*_\infty(LAT_2)$, as well as associated parameters. The Fibonacci and one-parameter lattice values are added as reference.

n	r_1	r_2	$d^*_\infty(LAT_2)$	$d^*_\infty(LAT_1)$	$d^*_\infty(Fib)$	Runtime (s)
15	0.07	0.843	0.1083	0.1233	0.1390	1.84
20	0.052	0.737	0.083795	0.0949	0.1188	11.21
25	0.0411	0.79167	0.0697	0.0774	0.095078	41.09
30	0.0343	0.784	0.05918	0.0694	0.079231	194.07
32	0.03218	0.72489	0.055749	0.0613	0.074279	731.00
34	0.03022	0.72497	0.052556	0.0578	0.069910	428.03
35	0.02936	0.72498	0.05107	0.0562	0.067913	1313.43
37	0.02777	0.72489	0.048303	0.0534	0.067861	1295.44
40	0.0256	0.725	0.0449	0.0496	0.063836	3158.23

5.3.3 Other Discrepancies

We describe in this subsection three possible reformulations of our models to build optimal sets for other discrepancy measures commonly used in discrepancy theory. This is not an exhaustive list of possible extensions of our models. We favored

constructions with the most common notions in the literature, or that could have an impact on practical applications of low-discrepancy sets.

5.3.3.1 Extreme Discrepancy

The extreme discrepancy removes the constraint that the lower-left corner of the box needs to be in $(0, 0)$, considering "axis-parallel" boxes rather than "anchored" ones. More formally, for a point set P,

$$D_\infty^{ext}(P) := \sup_{q,r\in[0,1)^d, q<r} \left| \frac{|P \cap [q, r)|}{|P|} - \lambda([q, r)) \right|,$$

where $q < r$ indicates that q is smaller than r for each coordinate. This discrepancy measure is more general than the L_∞ star discrepancy, but given that it is even more complicated to compute, it has been studied less. It is mentioned in [DGW14; Nie92], but there are only very few papers focusing explicitly on it such as [MC94]. Our models can be easily adapted for this change of discrepancy measure: it requires changing the lower bound of each sum in the open and closed box constraints (5.5a) and (5.5b) (model (5.5)) or (5.6a) and (5.6b) (model (5.6)), and considering the corresponding constraints for all relevant combinations of q and r. For example, in model (5.6), rather than always starting at 1 and going to i in each sum, we start at a certain $1 \leq k \leq i$. We once again use the general position hypothesis. While we do not provide a formal proof of this, and it is more a practical necessity, the results from Section 5.2.3 should transfer also to this case, with adapted definitions of admissible shits. Some care should also be put into checking that the box defined by (k, l) and (i, j) is valid, in other words that k is on the lower horizontal edge, i on the upper horizontal edge, l on the left vertical edge and j on the right vertical edge. In model (5.5) this can be checked using the indicator variables. In model (5.6), this is trivially done since we rely on the ordering of the x_i's and y_j's and do not check for critical boxes.

Adapting model (5.6) requires removing constraints (5.6a) and (5.6b) and replacing them by

$$\frac{1}{n}\sum_{u=k}^{i}\sum_{v=l}^{j} a_{uv} - (x_i - x_k)(y_j - y_k) \leq f \qquad \forall i, j, k, l = 0, \ldots, n, k \leq i, l \leq j \quad (5.9a)$$

$$\frac{-1}{n}\sum_{u=k+1}^{i-1}\sum_{v=l+1}^{j-1} a_{uv} + (x_i - x_k)(y_j - y_k) \leq f \quad \forall i,j,k,l = 0,\dots,n+1, k < i, l < j \tag{5.9b}$$

$$x_0 = y_0 = 0,\ a_{0j} = a_{i0} = 0 \qquad \forall i,j = 0,\dots,n. \tag{5.9c}$$

These are the only constraints that have an impact on the discrepancy value, constraints (5.6c) to (5.6g) are unchanged. Given that the set of boxes defining the L_∞ star discrepancy is a subset of those defining the extreme discrepancy, our lower bounds for the star discrepancy are also valid for the extreme discrepancy: constraint (5.6h) can be kept. However, constraints (5.6i) to (5.6p) proceeding from our results in Section 5.2.3 do not directly generalize to this case.[28] We note that we still use the ε spacing with a small constant for technical reasons.

For the extreme discrepancy, the model has many more constraints. It is therefore expected that we are not able to obtain solutions for n as high as in the L_∞ star discrepancy case. Table 5.8 describes the runtimes and discrepancy values obtained.

Table 5.8: Optimal discrepancies and runtimes for the L_∞ extreme discrepancy with an adapted Model (5.6). Values are rounded with 10^{-3} precision. The value in bold is correct up to 10^{-3} precision, but the exact value is between 0.20836 and 0.20845.

n	1	2	3	4	5
$D_\infty^{ext}(OPT_{ext})$	1	0.618	0.495	0.395	0.339
Runtime (s)	0.00	0.08	1.43	1.65	41.94
n	6	7	8	9	10
$D_\infty^{ext}(OPT_{ext})$	0.298	0.269	0.25	0.227	**0.208**
Runtime (s)	22.55	44.49	11 193	13 631	>500 000

5.3.3.2 Periodic Discrepancy

Another common aspect in discrepancy measures is to consider $[0,1)^d$ as a d-dimensional torus (here $d = 2$). The discrepancy is now defined by

$$D_\infty^{per}(P) := \sup_{q,r\in[0,1)^2} \left| \frac{|P \cap [q,r)|}{|P|} - \lambda([q,r)) \right|,$$

28 We note that we have simply not yet considered this question. It is quite likely that a general position result can also be expected here.

where each one-dimensional element composing $[q, r)$ is given by $[q_i, r_i)$ if $q_i \leq r_i$ and $[q_i, 1) \cup [0, r_i)$ otherwise. This notion is also known as "Weyl", "toroidal", or "wrap around" discrepancy and was studied by Lev in [Lev96]. We notice that this is a further generalization of the set of boxes defined for the extreme discrepancy. It is notably more difficult than the star discrepancy. Indeed, as was recently shown by Gilibert, Lachmann and Müllner in [GLM22], the VC-dimension of the class of boxes required to define it is in $O(d \log(d))$ and not simply $O(d)$. Using the notations of model (5.6), given a quadruplet (i, j, k, l) where $i > k$ and $j > l$, a box can be of one of the four following types:

- The regular extreme discrepancy box $[x_k, x_i) \times [y_l, y_j)$. The constraints are the same as (5.9a), (5.9b) and (5.9c).
- The box wrapping around the x-axis $([0, x_k) \cup [x_i, 1)) \times [y_l, y_j)$. The corresponding constraints are (5.10a) and (5.10b).
- The box wrapping around the y-axis $[x_k, x_i] \times ([0, y_l) \cup [y_j, 1))$. This is associated with constraints (5.10c) and (5.10d).
- The box wrapping around both axes $([0, x_k) \cup [x_i, 1)) \times ([0, y_l) \cup [y_j, 1))$. This corresponds to the two final constraints (5.10e) and (5.10f).

Since every box defining the discrepancy has to share its axes with the (x_i) and (y_j) we define, one can easily check that all candidate boxes are covered by one of the types above for a specific quadruplet (i, j, k, l) where $i > k$ and $j > l$.

By the arguments described above, the model for the periodic discrepancy can be obtained by taking that of the extreme discrepancy to which we add the following constraints. Once again, we are modifying model (5.6), but it is also possible to adapt model (5.5). We take the same constraints as for the extreme case, including the dummy variables, and add the following constraints.

$$\frac{1}{n}\sum_{v=l}^{j}\Big(\sum_{u=1}^{k} a_{uv}+\sum_{u=i}^{n} a_{uv}\Big)-(1-x_i+x_k)(y_j-y_l) \leq f \qquad \forall i, j, k=1,\dots,n,\ l=0,\dots,n,\ k<i, l<j \tag{5.10a}$$

$$\frac{-1}{n}\sum_{v=l+1}^{j-1}\Big(\sum_{u=1}^{k-1} a_{uv}+\sum_{u=i+1}^{n} a_{uv}\Big)+(1-x_i+x_k)(y_j-y_l) \leq f \qquad \forall i, j, k=1,\dots,n+1,\ l=0,\dots,n,\ k<i, l<j \tag{5.10b}$$

$$\frac{1}{n}\sum_{u=k}^{i}\Big(\sum_{v=1}^{l} a_{uv}+\sum_{u=j}^{n} a_{uv}\Big)-(x_i-x_k)(1-y_j+y_l) \leq f \qquad \forall i, j, l=1,\dots,n,\ k=0,\dots,n,\ k<i, l<j \tag{5.10c}$$

$$\frac{-1}{n}\sum_{u=k+1}^{i-1}(\sum_{v=1}^{l-1}a_{uv}+\sum_{v=j+1}^{n}a_{uv})+(x_i-x_k)(1-y_j+y_l)\leq f \qquad \forall i,j,l=1,\dots,n+1,\ k=0,\dots,n,\ k<i,l<j \tag{5.10d}$$

$$\frac{1}{n}(\sum_{u=1}^{k}\sum_{v=1}^{l}a_{uv}+\sum_{u=i}^{n}\sum_{u=j}^{n}a_{uv})-(1-x_i)(1-y_j)-x_k y_l\leq f \qquad \forall i,j,k,l=1,\dots,n,k<i,l<j \tag{5.10e}$$

$$\frac{-1}{n}(\sum_{u=1}^{k-1}\sum_{v=1}^{l-1}a_{uv}+\sum_{u=i+1}^{n}\sum_{v=j+1}^{n}a_{uv})+(1-x_i)(1-y_j)+x_k y_l\leq f \qquad \forall i,j,k,l=1,\dots,n+1,k<i,l<j \tag{5.10f}$$

Table 5.9 describes the run times and discrepancy values obtained. The periodic discrepancy model contains more constraints than the extreme one, and hence the maximal n for which we are able to find optimal point sets in reasonable time is further reduced.

Table 5.9: Optimal discrepancies and runtimes for the L_∞ periodic discrepancy.

n	1	2	3	4	5	6	7
$D_\infty^{per}(OPT)$	1	0.75	0.618	0.5	0.4249	0.3671	0.3279
Runtime	0.1	0.6	0.63	0.84	53.80	98.66	597.78

5.3.3.3 Multiple-corner Discrepancy

Unlike the two previous notions which are common in theoretical discrepancy literature, this notion is new and targets applications. The L_∞ star discrepancy creates an asymmetry by giving more importance to $(0,\dots,0)$. Often, it is undesirable to orient the regularity of our point set based on a specific corner of the space. To limit the impact of this, a possible counter-measure is to consider the L_∞ star discrepancy according to all 2^d corners, the worst of which is then the discrepancy value. In dimension 2, this *4-corner discrepancy* is defined by the following

$$d_\infty^{4c}(P) := \sup_{q=(q_1,q_2)\in[0,1)^2} \max\{D_{\text{loc}}([0,q),P), D_{\text{loc}}((q,1],P), D_{\text{loc}}([0,q_1)\times(q_2,1],P), D_{\text{loc}}((q_1,1]\times[0,q_2),P)\},$$

where $D_{\text{loc}}(q,P)$ is the local discrepancy of the box q for the point set P, c.f. Section 3.1.

For a given point set P, this can be reformulated w.r.t. the classical L_∞ star discrepancy by considering, in addition, the sets $P_2 := \{(1-x,y) : (x,y)\in P\}$, $P_3 := \{(1-x,1-y) : (x,y)\in P\}$ and $P_4 := \{(x,1-y) : (x,y)\in P\}$.

We then have

$$D_\infty^{4cor}(P) = \max(d_\infty^*(P), d_\infty^*(P_2), d_\infty^*(P_3), d_\infty^*(P_4)).$$

Given the definition above, the definitions of the point sets, and our models for the star discrepancy, it can be easily seen that we simply need to make four copies of the constraints (5.6a) and (5.6b) to adapt model (5.6) to this situation. In addition to the constraints for P given previously, the following constraints need to be added.

$$\frac{1}{n}\sum_{u=i}^{n}\sum_{v=1}^{j} a_{uv} - (1-x_i)y_j \le f \qquad \forall i,j = 1,\dots,n \qquad (5.11a)$$

$$\frac{-1}{n}\sum_{u=i+1}^{n}\sum_{v=1}^{j-1} a_{uv} + (1-x_i)y_j \le f \qquad \forall i,j = 1,\dots,n+1 \qquad (5.11b)$$

$$\frac{1}{n}\sum_{u=i}^{n}\sum_{v=j}^{n} a_{uv} - (1-x_i)(1-y_j) \le f \qquad \forall i,j = 1,\dots,n \qquad (5.11c)$$

$$\frac{-1}{n}\sum_{u=j+1}^{n}\sum_{v=i+1}^{n} a_{uv} + (1-x_i)(1-y_j) \le f \qquad \forall i,j = 1,\dots,n+1 \qquad (5.11d)$$

$$\frac{1}{n}\sum_{u=1}^{i}\sum_{v=j}^{n} a_{uv} - x_i(1-y_j) \le f \qquad \forall i,j = 1,\dots,n \qquad (5.11e)$$

$$\frac{-1}{m}\sum_{u=1}^{i-1}\sum_{v=j+1}^{n} a_{uv} + x_i(1-y_j) \le f \qquad \forall i,j = 1,\dots,n+1 \qquad (5.11f)$$

Finally, Table 5.10 gives the results obtained by our model with the multiple-corner discrepancy. We specify both the multiple-corner discrepancy of our set, and its actual L_∞ star discrepancy (necessarily smaller than or equal to it, as all the L_∞ star discrepancy constraints are included in the model). We note that the L_∞ star discrepancy values are not much higher than those of the optimal sets, and far better than previously known sets. This suggests that considering a set with good multiple-corner discrepancy could be a good tradeoff for applications, without losing much for the L_∞ star discrepancy.

29 We have not yet been able to prove, or disprove, this result.

Table 5.10: Comparison of previously best values for low-discrepancy sets and our optimal sets, both the 4-corner optimal set OPT_{4c} and the L_∞ star discrepancy optimal set OPT_{star}. We note that the optimal 4-corner sets have the same L_∞ star discrepancy and 4-corner discrepancy in all tested cases.[29]

n	3	4	5	6	7	8	9	10
$D^*_\infty(OPT_{star})$	0.2847	0.2500	0.2000	0.1667	0.1500	0.1328	0.1235	0.1111
$D^*_\infty(Fib)$	0.5880	0.4910	0.3528	0.3183	0.2728	0.2553	0.2270	0.2042
$D^*_\infty(OPT_{4c})$	0.3333	0.2548	0.2166	0.1875	0.1632	0.1438	0.1319	0.1197
$D^{4cor}_\infty(OPT_{4c})$	0.3333	0.2548	0.2166	0.1875	0.1632	0.1438	0.1319	0.1197
Runtime	0.02	0.12	0.3	0.68	1.14	1.97	4.34	12.68

n	11	12	13	14	15	16	17	18
$D^*_\infty(OPT_{star})$	0.1030	0.0952	0.0889	0.0837	0.0782	0.0739	0.06996	0.06667
$D^*_\infty(Fib)$	0.1857	0.1702	0.1571	0.1459	0.1390	0.1486	0.1398	0.1320
$D^*_\infty(OPT_{4c})$	0.1083	0.09999	0.09502	0.08874	0.08429	0.07916	0.07510	0.0715
$D^{4cor}_\infty(OPT_{4c})$	0.1083	0.09999	0.09502	0.08874	0.08429	0.07916	0.07510	0.0715
Runtime	26.45	46.46	88.46	219.22	783.74	1616.65	3287.64	7412.73

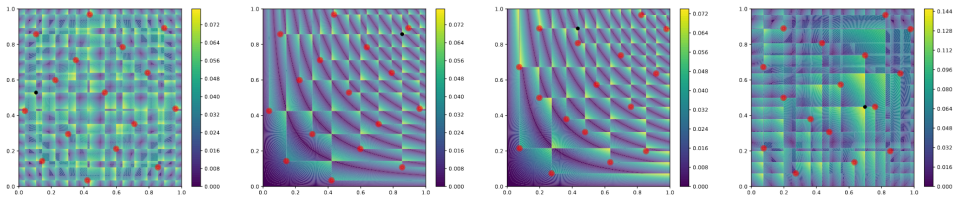

Figure 5.10: n=16. Left to right: local 4-corner discrepancy of the optimal 4-corner set, local L_∞ star discrepancy of the optimal 4-corner set, local star discrepancy of the optimal L_∞ star set and local 4-corner discrepancies of the optimal L_∞ star set. While the optimal 4-corner set is very good for the L_∞ star discrepancy, the same can't be said for the 4-corner discrepancy of the optimal L_∞ star set.

5.4 Conclusion

In this chapter, using two types of non-linear programming models, we provide optimal constructions for sets of minimal L_∞ star discrepancy for $n = 1$ to $n = 21$ in two dimensions and $n = 1$ to $n = 8$ in three dimensions. For $n \geq 7$ in dimension 2, these sets have much lower discrepancy than the previously known sets. Analyzing the local discrepancies of these sets and comparing to known low-discrepancy sequences also shows a clear structural difference, suggesting that known sequences are over-sampling in the lower zones of $[0, 1]^2$ compared to suggest that new approaches are possible to design low-discrepancy sets and

sequences. We furthermore observed an almost negligible extra cost for point sets that are well-distributed not only for the origin $(0, \ldots, 0)$ but also other corners of the square.

Apart from the conjectures mentioned in the chapter such as higher dimension general position, there are many further research questions that deserve being looked into. Naturally, finding heuristic approaches or improving our models to find excellent solutions for higher n should be the first goal. Though preceding this work chronologically, the two following chapters on subset selection provide a first method of computing better sets in higher dimensions. We will also introduce greedy algorithms for the L_∞ and the L_2 discrepancy in Chapter 8, greatly inspired by our approaches here. The possibility of constructing block by block a low-discrepancy set has not been considered so far and is promising. For example, we are able to create an excellent 16 point set by adding 8 points to an existing 8 point set. Overall, we believe this to be a first step in a promising new direction for the discrepancy community as very little had been done so far to harness the power of optimization formulations, barring Thiémard's [Thi01b], and Doerr and de Rainville's [DR13] work.

6 Subset Selection: Exact approaches

This chapter follows [CDP22], with a few complementary additions: a feasibility approach, the L_2 version of the L_∞ star discrepancy subset selection problem and the description of an easier case. This is joint work with Carola Doerr and Luís Paquete.

6.1 Summary of Results

6.1.1 Motivation

Given the advantageous behavior of point sets of small discrepancy in practice[30], we study in this chapter how to choose from a given set P of n points a subset P_k of size $k \leq n$ such that the L_∞ star discrepancy of P_k is minimized. This *star discrepancy subset selection problem* (SDSSP) has its origins in Machine Learning (ML) and in optimization, and in particular in the instance selection problem, where one aims to select from a given set of instances a small subset that maximizes diversity – with the idea that more diverse instances provide better training opportunities for ML-based approaches. An example for such an approach can be found in [Neu+18], where diverse images and instances of the traveling salesperson problem (TSP) are constructed via an evolutionary approach. In each iteration, the evolutionary algorithm generates a set of new instances and a selection operator then updates by selecting instances from the old and the newly generated ones. Since no efficient algorithms were known in [Neu+18] to address the general star discrepancy subset selection problem, only so-called "+1" schemes are considered, which generate only one new instance per iteration. As well as practical applications, we aim to obtain excellent point sets for (n, d) combinations that could not be reached with the optimal constructions in the previous chapter.

The problem of selecting subsets with respect to small discrepancy values was also the focus of the work on so-called *online thinning*, presented in [Dwi+19]. Online thinning requires a decision maker to either accept or reject a point of a sequence into a selected subset, with the goal of minimizing the discrepancy of the selected set. The process studied in [Dwi+19] assumes, in addition, that at least one

30 We refer to some of the examples in the introduction such as [Bou+17] or [Cau+20], but this is not an exhaustive list.

out of every two consecutive points has to be selected. The three main differences between their work and ours are: (1) while *sequences* are studied in [Dwi+19], we consider fixed point *sets* P and a fixed target size k, (2) we optimize over all possible subsets of a given size k, and (3) in contrast to [Dwi+19], our approaches are not restricted to uniformly sampled i.i.d. points.

6.1.2 Our Contribution

Previous results on the NP-hardness of calculating the star discrepancy [GSW09] hint to the difficulty of solving this problem exactly. We show by a reduction from the DOMINATING-SET problem that the decision version of the star discrepancy subset selection problem is NP-hard. We then study the efficiency of algorithmic approaches for the star discrepancy subset selection problem. Simple algorithmic approaches such as random subset selection and iterative greedy selection do not perform well, motivating the design and the analysis of a mixed-integer linear formulation as well as a combinatorial branch-and-bound approach for this problem, introduced in Section 6.3. The mixed-integer linear formulation (MILP) is a natural formulation of the discrepancy subset selection problem that uses the grid structure of the star discrepancy introduced in Section 3.1. Our branch and bound (BB) is a classical approach that starts from a greedy solution and uses combinatorial lower bounds for pruning, which can be computed in an incremental manner, also relying on the grid formulation in equation (3.4). Our experimental results in Section 6.4 for $d = 2$ indicate that BB presents better performance for small k/n ratios while MILP performs better for large k/n ratios. We relate these findings to the quality of the lower bounds of MILP. Unfortunately, the performance deteriorates strongly already for $n > 140$ and for $d > 3$, so that we have to restrict our analysis to the two- and three-dimensional cases.

As a side result, we observe that subset selection can be an interesting approach to generate point sets of small discrepancy values. For our two-dimensional test cases, the star discrepancy of the best found size-k subsets of the Sobol', the Faure, the Halton, and the reverse Halton [VC06] sequences are around 50% smaller than the star discrepancy of the original construction of the same size for $k = 20$ and 40. For larger k, the advantage is slightly smaller, but still 40%, on average, for $k = 60$, 36% for $k = 80$, and 44% for $k = 100$. Similar advantages are obtained in the three-dimensional case for these four sequences. Much better advantages of at least 60% are obtained for uniform samples for $d = 2$ and $d = 3$ and for Latin Hypercubes for $d = 3$. For the Fibonacci set for $d = 2$, in contrast, the advantages are much less important, it is less than 1% for $k = 80$ and $k = 100$, but it is slightly above 27% and 22% for $k = 20$ and $k = 40$, respectively. The point sets with the

best star discrepancy for each value of k in the two-dimensional case that were obtained in our experiments are available at https://algo.dei.uc.pt/star. The BB code for $d = 2$ is available at https://github.com/luis-paquete/StarDSS.

In complement, this chapter provides three extra approaches relative to [CDP22]. Section 6.3.4 describes a feasibility approach to the MILP formulation. Rather than trying to find the best solution from scratch, we fix a discrepancy objective and attempt to obtain a subset that reaches this value, or lower. While it is able to solve the problem very quickly in many cases, this approach struggles a lot more when the target value is near, or slightly below, the real optimal solution.[31] Section 6.3.5 underlines an easier case, when points can be ordered with a dominance relation. While not too interesting per se, it could provide a good basis should good decomposition methods be found for the subset selection problem.

Finally, we round out this chapter on exact methods for subset selection with a possible approach for the L_2 subset selection in Section 6.7. While very different to the L_∞ methods, it has the advantage of *being linearly dependent on the dimension.* Recent success with L_2 techniques to construct good L_∞ sets could rekindle interest in tackling this question more precisely. We note in particular that it can be expressed as an Unconstrained Binary Quadratic Problem, for which numerous algorithms and heuristics exist.

6.2 The Star Discrepancy Subset Selection Problem

Given a d-dimensional point set $P \subseteq [0,1]^d$ of size $|P| = n$, and given an integer $k \leq n$, the goal is to find a subset $P^* \subseteq P$ of size $|P^*| = k$ such that $d^*_\infty(P^*)$ is minimized. Using equation (3.4), the L_∞ *star discrepancy subset selection problem* has the following equivalent formulation:

$$\min_{\substack{P^* \subseteq P \\ |P^*|=k}} \max \left\{ \max_{q \in \overline{\Gamma}(P^*)} \delta(q, P^*), \max_{q \in \Gamma(P^*)} \overline{\delta}(q, P^*) \right\}. \tag{6.1}$$

We show NP-hardness of this problem in Section 6.2.1 and we discuss some basic properties in Section 6.2.2.

31 Once again, this matches our comment on all optimization approaches for discrepancy: finding an upper bound is easy, finding a good lower bound is much harder.

6.2.1 NP-Hardness of the Subset Selection Problem

We consider the following decision version of the L_∞ star discrepancy subset selection problem:

Decision problem: Discrepancy Subset Selection
Instance: natural numbers $n, k \in \mathbb{N}$, $k \leq n$, $\varepsilon \in (0, 1]$, point set $P = (x^{(i)})_{i\in\{1,\dots,n\}}$
Question: Is there $P' \subseteq P$ such that $|P'| = k$ and $d^*_\infty(P') \leq \varepsilon$?

▶ **Theorem 6.1.** The decision version of the discrepancy subset selection problem is NP-hard. ◀

As in the proof of the NP-hardness of calculating the L_∞ star discrepancy presented in [GSW09], we will obtain Theorem 6.1 by a reduction from the DOMINATING-SET problem, a problem that is well known to be NP-complete (see, for example, [GJ90]).

Decision problem: DOMINATING-SET
Instance: Graph $G = (V, E)$, $k \in \{1, \dots, |V|\}$
Question: Is there a set $T \subseteq V$ of size at most k such that for any $v \in V \setminus T$, there exists $t \in T$ such that $(v, t) \in E$?

Proof of Theorem 6.1. Throughout this proof, let $q \in [0, 1]^n$. We consider an instance $G = (V, E)$, $k \in \{1, \dots, |V|\}$ of DOMINATING-SET. We build a point set $P = (x^{(i)})_{i\in\{1,\dots,n\}}$ in $\mathbb{R}^n$ where $n = |V|$ by defining, for all $i, j \in \{1, \dots, n\}$,

$$x^{(i)}_j := \begin{cases} \alpha & \text{if } (i, j) \in E \text{ or } i = j, \\ \beta & \text{otherwise,} \end{cases}$$

where α and β are two real values such that $\frac{1}{n} > \alpha > \beta > 0$.

To begin, we introduce the following formula for the L_∞ star discrepancy, shown in [GSW09]. For any point set P,

$$d^*_\infty(P) = \max\Big\{ \max_{\ell=0,\dots,n-1} (V^\ell_{\max} - \tfrac{\ell}{n}), \max_{\ell=1,\dots,n} (\tfrac{\ell}{n} - V^\ell_{\min}) \Big\}, \tag{6.2}$$

where $V^\ell_{\min}$ is the volume of the smallest (by the Lebesgue measure) closed box containing at least ℓ elements of P, and $V^\ell_{\max}$ is the volume of the largest half-open box containing at most ℓ elements of P.

We will set aside the case $\ell = n$ in the first part of the proof. We consider P_T, a subset of P of size k associated to a subset $T \subseteq V$ of size k. For this point set P_T, the largest empty box $V^0_{\max}$ is of size at least β^n and at most α^n. Since the maximal coordinate for any point in P_T is α, any half-open box that does not contain all the

points of P_T will have at least one coordinate smaller than α. This gives us $V^{\ell}_{\max} \leq \alpha$ for all $\ell \in \{1, \ldots, n-1\}$. We obtain the upper-bound α for the first maximum in (6.2) by choice of α and β.

For the second maximum, any closed box containing at least one point (but not all of them) will have each coordinate greater than or equal to β since the lowest coordinate in any dimension for each point is β. This gives us a minimum volume of β^n for a box containing some points of P_T. The fraction $\frac{\ell}{n}$ can be upper-bounded by $\frac{n-1}{n}$, which gives us an upper bound of $\frac{n-1}{n} - \beta^n \geq \max_{\ell=1,\ldots,n-1}(\frac{\ell}{n} - V^{\ell}_{\min})$.

We now consider the case $\ell = n$ depending on whether or not T is a dominating set of G. If T is a dominating set, by our point set construction, for any $j \in \{1, \ldots, n\}$, there exists $i \in T$ such that $x^{(i)}_j = \alpha$. Any box of the type $[0, q] = \prod_{i=1}^{n}[0, q_i]$ will contain all the points of P_T if and only if for all $i \in \{1, \ldots, n\}, q_i \geq \alpha$. Therefore, the smallest box containing all the elements of P_T has volume α^n. This gives us a $1 - \alpha^n$ term in (6.2), which is greater than all the other terms calculated previously, by choice of α and β.

If T is not a dominating set, there exists a vertex i not dominated by the elements of T. Since i is not dominated by T, the smallest full-box has size at most $\alpha^{n-1}\beta$ since all the points in P_T have β as their i-th coordinate. This gives us at least a $1 - \alpha^{n-1}\beta$ term in equation (6.2) which like in the previous case is also greater than all the other terms. It is also strictly greater than $1 - \alpha^n$. In both cases, we have shown that the discrepancy value is obtained by the volume of the smallest full-box. We have that $d^*_\infty(P_T) \leq 1 - \alpha^n$ if and only if T is a dominating set of G. We note that any dominating set T of size strictly smaller than k can become a dominating set of size exactly k by adding points from G until T is of size exactly k. This gives us the desired result: G has a dominating set of size at most k if and only if P has a subset of size k of discrepancy at most $1 - \alpha^n$. ■

We note that while the problem is NP-hard, it is not NP-complete to our knowledge. If we are given a subset of a point set P, checking if its discrepancy is smaller than ϵ cannot be done in polynomial time, under the hypothesis that P ≠ NP. This comes from the fact that we want an upper bound on the discrepancy and not a lower bound. For a lower bound, given a specific anchored box, we can verify that the discrepancy is large enough in linear time by counting the points in the box and calculating its volume.[32] On the other hand, for an upper bound, we would need to check that all the possible anchored boxes have a small enough local discrepancy. It is not sufficient to exhibit one of them.

32 As mentioned in Section 3.2, the star discrepancy calculation was shown to be NP-complete in [GSW09].

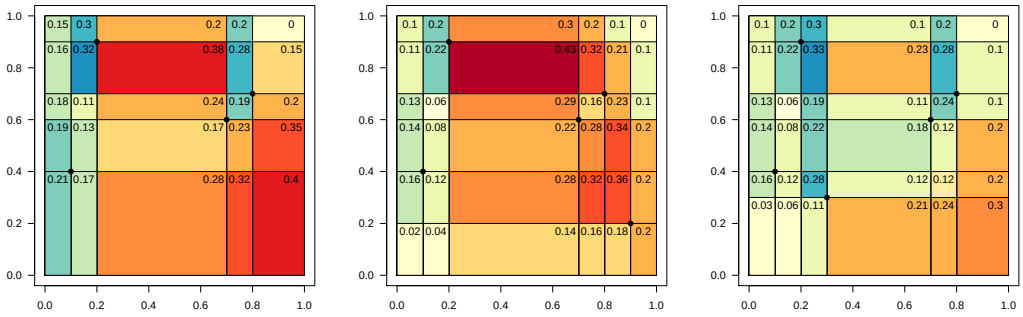

Figure 6.1: Illustration of the point sets defined in Example 6.2. The values are the local star discrepancy values and the color of each cell is the local star discrepancy value of its upper right corner (darker colors are used for larger discrepancy values). Blue colors are used when $d^*_\infty(y,P) = \overline{\delta}(y,P)$, and red colors are used when $d^*_\infty(y,P) = \delta(y,P)$.

6.2.2 Other Basic Properties of the Discrepancy Subset Selection Problem

Non-Monotonic Behavior of the Star Discrepancy Before we discuss our strategies to solve the star discrepancy subset selection problem, we first note that the star discrepancy is a non-monotone function, in the sense that $P' \subseteq P$ does not imply any order of $d^*_\infty(P')$ and $d^*_\infty(P)$. The following example illustrates this non-monotonic behavior. It is visualized in Figure 6.1.

▶ **Example 6.2.** Let $P := \{(0.1, 0.4), (0.2, 0.9), (0.7, 0.6), (0.8, 0.7)\}$. Then, it holds that $d^*_\infty(P) = 0.40$, whereas $d^*_\infty(P \cup \{(0.9, 0.2)\}) = 0.43$ and $d^*_\infty(P \cup \{(0.3, 0.3)\}) = 0.33$. As we can see in Figure 6.1, the discrepancy value of the first set is determined by the point $q = (1.0, 0.4)$, whereas it is determined by points $(0.7, 0.9)$ and $(0.3, 0.9)$ in the second and third case, respectively. ◀

Non-Monotonic Behavior of the Subset Selection Problem For a fixed set of points P (therefore, fixed n), there is also no relation between the smallest discrepancy subsets obtained for different k. To illustrate this non-monotonic behavior, consider the following simple example in dimension 1. Let $P := \{\frac{1}{6}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{5}{6}\}$. Niederreiter introduced in [Nie72] an explicit formula for the discrepancy in dimension 1, also showing that for an n-point set, the minimal discrepancy is uniquely obtained by the set $\{\frac{1}{2n}, \frac{3}{2n}, \ldots, \frac{2n-1}{2n}\}$. The optimal subset of size 2 of P is therefore given by $P'_2 = \{\frac{1}{4}, \frac{3}{4}\}$, whereas the optimal subset of size 3 is given by $P'_3 = \{\frac{1}{6}, \frac{1}{2}, \frac{5}{6}\}$.

Another example for this non-monotonic relationship between optimal subsets of different sizes can be found in the results presented in this chapter. For example, in the 2-dimensional setting, the best size $k = 20$ subset of the first $n = 140$ elements of the Fibonacci set is not contained in the best subset of size $k = 120$.[33]

Extending the Grid to its Original Size For the design of our mixed-integer linear formulation, it will be convenient to consider the whole grid induced by P, and not only the one induced by P^*.

▶ **Lemma 6.3.** The star discrepancy subset selection problem is equivalent to the following:

$$\min_{\substack{P^* \subseteq P \\ |P^*|=k}} \max \left\{ \max_{q \in \overline{\Gamma}(P)} \delta(q, P^*), \max_{q \in \Gamma(P)} \overline{\delta}(q, P^*) \right\}. \tag{6.3}$$

◀

Proof. Let $P^* \subseteq P$, $|P^*| = k$. We show that $\max_{q \in \overline{\Gamma}(P)} \delta(q, P^*) = \max_{q \in \overline{\Gamma}(P^*)} \delta(q, P^*)$ and that $\max_{q \in \Gamma(P)} \overline{\delta}(q, P^*) = \max_{q \in \Gamma(P^*)} \overline{\delta}(q, P^*)$. Since $P^* \subseteq P$, we only need to prove "$\leq$". To show the first equation, let $q \in \overline{\Gamma}(P)$. For every coordinate $j \in \{1, \ldots, d\}$, let $u_j := \min\{u \in \overline{\Gamma}_j(P^*) \mid u \geq q_j\}$. Then $D(u, P^*) = D(q, P^*)$ and hence $\delta(u, P^*) = \lambda(u) - \frac{1}{|P^*|} D(u, P^*) \geq \lambda(q) - \frac{1}{|P^*|} D(q, P^*) = \delta(q, P^*)$. This shows that $\max_{q \in \overline{\Gamma}(P)} \delta(q, P^*) \leq \max_{q \in \overline{\Gamma}(P^*)} \delta(q, P^*)$.

For the second equation, let $q \in \Gamma(P)$. Set $\ell_j := \max\{\ell \in \Gamma_j(P^*) \cup \{0\} \mid \ell \leq q_j\}$. Then $\overline{D}(\ell, P^*) = \overline{D}(q, P^*)$ and thus $\overline{\delta}(\ell, P^*) = \frac{1}{|P^*|} \overline{D}(\ell, P^*) - \lambda(\ell) \geq \frac{1}{|P^*|} \overline{D}(q, P^*) - \lambda(q) = \overline{\delta}(q, P^*)$. ■

6.3 Algorithmic Approaches to Solve the Discrepancy Subset Selection Problem

In this section, we suggest two different approaches to solve the star discrepancy subset selection problem, one based on mixed-integer linear programming (MILP), Section 6.3.1, and one based on branch and bound, Section 6.3.2. In Section 6.4, we compare their performance against two heuristics, random subset selection and an iterative greedy selection, which we use to obtain an initial solution for the branch and bound algorithm. The greedy approach is described in Section 6.3.3. Finally,

33 We do not discuss the position of the points in these optimal subsets here in this work, but the sets can be found in the repository available at https://algo.dei.uc.pt/star.

we present a feasibility approach in Section 6.3.4, which performed similarly to the MILP formulation it is based on, and describe one case where subset selection is easy to solve in Section 6.3.5.

Convention: To ease the description of our algorithms, we assume that, for all $j \in \{1, \dots, d\}$, the coordinates $\{p_j^{(i)} \mid i \in \{1, \dots, n\}\}$ are pairwise different.

6.3.1 A Mixed Integer Linear Programming Formulation

For simplification purposes, we start with the description of the mixed integer linear programming model for the star discrepancy subset selection problem for $d = 2$. We then discuss extensions for larger dimensions.

The following component-wise order relations in $\mathbb{R}^d$ will be required for our model. For $v, w \in \mathbb{R}^d$, we write

$$\begin{aligned} v \leqq w &\iff v_j \leq w_j \text{ for all } j \in \{1, \dots, d\} \\ v \leq w &\iff v \neq w \text{ and } v \leqq w \\ v < w &\iff v_j < w_j \text{ for all } j \in \{1, \dots, d\} \end{aligned}$$

Consider a two-dimensional point set $P := \{p^{(1)}, p^{(2)}, \dots, p^{(n)}\} \subseteq \mathbb{R}^2$.[34] Without loss of generality, we assume the points in P are reordered such that $p_1^{(1)} \leq p_1^{(2)} \leq \cdots \leq p_1^{(n)}$. Let $\mathcal{S}_n$ denote the symmetric group of order n and $\sigma \in \mathcal{S}_n$ denote a permutation of $\{1, \dots, n\}$, such that $p_2^{(\sigma(1))} \leq p_2^{(\sigma(2))} \leq \cdots \leq p_2^{(\sigma(n))}$.[35]

We shall use $\gamma_{i,j}(P)$ to denote the grid point at position (i, j) in $\overline{\Gamma}(P)$, $i, j \in \{1, \dots, n+1\}$.[36] Then, due to the ordering of the points in P, it holds that $\gamma_{i,\sigma(i)}(P) = p^{(i)}$, $\gamma_{i,n+1}(P) = (p_1^{(i)}, 1)$, $\gamma_{n+1,\sigma(i)}(P) = (1, p_2^{(\sigma(i))})$, for $i \in \{1, \dots, n\}$, and $\gamma_{n+1,n+1}(P) = (1, 1)$. In addition, we define the following index sets

$$\begin{aligned} \overline{\Delta}(P, i, j) &:= \{\ell \mid p^{(\ell)} \leqq \gamma_{i,j}(P), p^{(\ell)} \in P\} \qquad \text{and} \\ \Delta(P, i, j) &:= \{\ell \mid p^{(\ell)} < \gamma_{i,j}(P), p^{(\ell)} \in P\} \end{aligned}$$

for $i, j \in \{1, \dots, n+1\}$.

For our MILP model, we define a binary variable x_i that takes value 1 if point $p^{(i)}$ is selected, $i \in \{1, \dots, n\}$, and 0 otherwise. The model is as follows.

34 For this model, the points of the set are $p^{(i)}$, while variables of the model will be given by x_i.

35 Where our convention of pairwise different coordinates does not apply, we assume the following: In the case of a tie $p_1^{(i)} = p_1^{(i+1)}$ in the first coordinate, we assume that $p_2^{(i)} \leq p_2^{(i+1)}$. In the case of a tie in the second coordinate $p_2^{(\sigma(i))} = p_2^{(\sigma(i+1))}$ we assume that that $p_1^{(\sigma(i))} \leq p_1^{(\sigma(i+1))}$.

36 For simplicity, we restrict the presentation to $\overline{\Gamma}(P)$, since $\Gamma(P) \subset \overline{\Gamma}(P)$.

$$
\begin{aligned}
\min \quad & z \\
\text{s.t.} \quad & z \geq h_{i,j} - \frac{1}{k} \sum_{\ell \in \Delta(P,i,j)} x_\ell && \text{for all } i,j \in \{1,\ldots,n+1\} \\
& z \geq -h_{i,j} + \frac{1}{k} \sum_{\ell \in \overline{\Delta}(P,i,j)} x_\ell && \text{for all } i,j \in \{1,\ldots,n\} \\
& \sum_{i=1}^{n} x_i = k \\
& x_i \in \{0,1\} && \text{for all } i \in \{1,\ldots,n\} \\
& z \in \mathbb{R}_{\geq 0}
\end{aligned}
\tag{6.4}
$$

Variable z is a non-negative continuous variable that takes the optimal star discrepancy value. The first two constraints are due to the linearization of the objective function in Problem (6.3) and bound the minimum value of z, where $h_{i,j}$ is the measure of the d-dimensional box $[0, \gamma_{i,j}(P)]$. The third constraint ensures that exactly k points in P are selected.

Extending our MILP for more dimensions, we obtain $O(n^d)$ constraints and $O(n)$ variables. Noteworthy, its relaxation, that is, $x_i \in [0,1]$, has an integral solution when $k = n$. This suggests that the *integrality gap*, i.e., the difference between the optimal value z^* for the original MILP (satisfying that there exists $x^* \in \{0,1\}^n$ such that the conditions are satisfied) and the optimal value $z^*_{\text{relax.}}$ of its relaxation (where we only require existence of $x \in [0,1]^n$ for which the conditions are satisfied), may be small when the ratio k/n is large. This suggestion is confirmed in the experimental results reported in Section 6.4.

6.3.2 A Combinatorial Branch-and-Bound Algorithm

Algorithm 1 presents the pseudocode of our combinatorial branch-and-bound approach (BB) for the star discrepancy subset selection problem, for a given k and a given point set P. At a given iteration, the algorithm maintains three stacks: S_A, which stores the points that were accepted (subset P_A), S_B, which stores the points that were rejected (subset P_R), and S_N, which stores the points for which a decision has not yet been taken (subset P_N). Both S_A and S_R are empty in the beginning, whereas S_N contains all points in P. Variable *ub* corresponds to the lowest upper bound on the optimal discrepancy value found so far and is initially set to 1, which clearly is an upper bound for the star discrepancy of any subset of P, since it is an upper bound for the star discrepancy of *any* point set.

Algorithm 1: Branch and Bound

$S_A := \emptyset$, $S_R := \emptyset$, $S_N := (p^{(1)}, ..., p^{(n)})$, $ub := 1$

Function $BB(S_A, S_R, S_N)$

1: **if** $|P_A| = k$ **then**
2: $ub := \min\{ub, d^*_\infty(P_A)\}$
3: **return**
4: **else if** $P_N = \emptyset$ or $|P_A| + |P_N| < k$ **then**
5: **return**
6: **else if** $LB(P_A, P_R, P_N) > ub$ **then**
7: **return**
8: **else**
9: $q := \text{pop}(S_N)$
10: $\text{push}(q, S_A)$
11: $BB(S_A, S_R, S_N)$
12: $p := \text{pop}(S_A)$
13: $\text{push}(p, S_R)$
14: $BB(S_A, S_R, S_N)$
15: $q := \text{pop}(S_R)$
16: $\text{push}(q, S_N)$
17: **return**

The branching part of the algorithm works as follows: at each recursive step, the point p at the top of stack S_N is removed and is placed at the top of stack S_A (p is accepted). Then, the remaining smaller sub-problems are solved recursively, with the points in S_A (and none of the points of S_R) belonging to the solutions of these sub-problems. When back to the same recursion level, the point p is removed from the top of stack S_A and placed at the top of stack S_R (p is rejected) and the same procedure is repeated again for the smaller sub-problems.

The usual stopping conditions avoid the generation of infeasible solutions, namely, either having k points or there are not enough points in S_N to reach a solution with k points. In the former case, the star discrepancy value of the k points is computed and compared against the upper bound (ub), which is updated accordingly. The function $LB(\cdot)$ allows the pruning of the search tree by computing a lower bound on the smallest value of star discrepancy of a feasible solution that contains the points stored in S_A. The following section describes the lower bound computations.

6.3.2.1 Lower Bounds

Consider that, at a given moment of Algorithm 1, stacks S_A, S_R, and S_N contain point sets P_A, P_R, and P_N, respectively. Note that $P_N := P \setminus (P_A \cup P_R)$. Let P_A^* be the set of k points with the smallest value of star discrepancy that contains P_A and does not intersect P_R, that is,

$$P_A^* := \arg\min\{d_\infty^*(P') \mid P_A \subseteq P' \subseteq P \setminus P_R, |P'| = k\}. \tag{6.5}$$

Our bounding function is the maximum of two values, that is,

$$LB(P_A, P_R, P_N) = \max\{LB_1(P_A, P_R, P_N), LB_2(P_A, P_R, P_N)\}.$$

The second value, $LB_2(P_A, P_R, P_N)$, is a lower bound on the local discrepancy of points in $\Gamma(P_A)$,

$$LB_2(P_A, P_R, P_N) := \max_{q \in \Gamma(P_A)} \left\{ \frac{1}{k} \overline{D}(q, P_A) - \lambda(q) \right\} \leq d_\infty^*(P_A^*). \tag{6.6}$$

Note that $\overline{D}(q, P_A) \leq \overline{D}(q, P_A^*)$ holds for every point $q \in \Gamma(P_A^*)$. This lower bound holds as any box $q \in \Gamma(P_A^*)$ will contain at least the points in $\Gamma(P_A)$.

The first value, $LB_1(P_A, P_R, P_N)$, is also a lower bound on the local discrepancy of points in $\overline{\Gamma}(P_A)$. For a given set P_A and set P_N, at each point q in $\overline{\Gamma}(P_A)$, an upper bound on the value of $D(q, P_A^*)$ is as follows

$$\min\{k, D(q, P_A) + D(q, P_N)\} \geq D(q, P_A^*) \tag{6.7}$$

from which the following lower bound $\eta(q, P_A, P_N)$ on the value of the local discrepancy at point q can be derived:

$$\eta(q, P_A, P_N) := \lambda(q) - \frac{1}{k} \min\{k, D(q, P_A) + D(q, P_N)\} \leq \delta(q, P_A^*). \tag{6.8}$$

One can see this lower bound as being the local discrepancy value for q should we pick the remaining points optimally *with respect to* q. Finally, by taking the worst of all these lower bounds, we define our lower bound $LB_1(P_A, P_R, P_N)$ as follows

$$LB_1(P_A, P_R, P_N) := \max_{q \in \overline{\Gamma}(P_A)} \{\eta(q, P_A, P_N)\} \leq d_\infty^*(P_A^*). \tag{6.9}$$

We already point out the main weakness of our lower bounds: we are overly optimistic for too many boxes at the same time, as we have no guarantee on which

points will be picked in the future. For small discrepancy values, for example around $3/k$, this means that we cannot stop a branch exploration until either we have placed too many points inside a box very quickly (relatively rare) or there are very few candidate points that could be added inside a box. In this last more common case, this usually means that we have explored nearly all the possible subsets that can be obtained from P_A. In other words, our lower bounds perform well for boxes with a small coordinate in the first dimension, but provide very little information for other boxes.

6.3.2.2 Lower Bound Computation

A naïve computation of $LB(P_A, P_R, P_N)$ requires $O(dk^{d+1})$ time for each new point p that enters into P_A due to the need to compute $D(q, P_A)$ and $D(q, P_N)$ for every point q in $\overline{\Gamma}(P_A)$ as well as $\overline{D}(q', P_A)$ and $\overline{D}(q', P_N)$ for every point q' in $\Gamma(P_A)$. However, note that $D(q, P_N)$ and $\overline{D}(q', P_N)$ can be pre-computed for every point $q \in \overline{\Gamma}(P)$ and every point $q' \in \Gamma(P)$, respectively, and for every subset P_N, which requires $\Theta(n^{d+1})$ space and time in the pre-processing step. Moreover, for each new point p that enters into P_A at each recursive step, $D(q, P_A)$ and $\overline{D}(q', P_A)$ need only to be computed at points $q \in \overline{\Gamma}(P_A)$ and $q' \in \Gamma(P_A)$ such that $p \preceq q$ and $p \preceq q'$ holds, respectively. In the following, we show that both components of the lower bound $LB(P_A, P_R, P_N)$ can be computed even faster in practice by considering a given ordering of the points in P.

As done in Section 6.3.1 and assuming that the points have pairwise different coordinates (as per our convention made at the beginning of this section), let us assume, without loss of generality, that $p_1^{(i)} < p_1^{(j)}$ holds for $i < j$, $i \in \{1, \ldots, n-1\}$. For this reason, for a given P_A and P_N, it holds that $\overline{D}(q, P_N) = 0$ and $D(q, P_N) = 0$ for every point $q \in \Gamma(P_A)$. In the following, we discuss particular properties that arise from this ordering and that will lead to an incremental evaluation of both lower bounds.

For a given P_A and P_N, let $p \in P_N$ be the smallest point with respect to the ordering above. We consider the following subsets of $\overline{\Gamma}(P_A \cup \{p\})$:

$$\overline{G}_0(p, P_A) := \overline{\Gamma}(P_A \cup \{p\}) \setminus \overline{\Gamma}(P_A) \qquad \text{and} \qquad \overline{G}_1(p, P_A) := \left\{ q \in \overline{\Gamma}(P_A) \mid p < q \right\}.$$

Note that due to the ordering of the points in P, we have that $\overline{G}_1(p, P_A)$ contains only points in $\overline{\Gamma}(P_A)$ that strictly dominate point p and have 1 in the first coordinate.

Update of LB_1 We state the following propositions for the incremental computation of $LB_1(P_A, P_R, P_N)$ in the case of inserting point p into P_A and into P_R, respectively.

▶ **Proposition 6.4.**

$$LB_1(P_A \cup \{p\}, P_R, P_N \setminus \{p\}) = \max\begin{cases} LB_1(P_A, P_R, P_N) \\ \max\limits_{q \in \overline{G}_0(p, P_A)} \{\eta(q, P_A \cup \{p\}, P_N \setminus \{p\})\} \end{cases}$$

◀

Proof. We prove that for every point q in $\overline{\Gamma}(P_A)$, it holds that $\eta(q, P_A, P_N) = \eta(q, P_A \cup \{p\}, P_N \setminus \{p\})$ and, therefore, only the points in $\overline{G}_0(p, P_A)$ need to be considered. For this, we partition $\overline{\Gamma}(P_A)$ in two disjoint subsets, $\overline{\Gamma}(P_A) \setminus \overline{G}_1(p, P_A)$ and $\overline{G}_1(p, P_A)$.

i) If $q \in \overline{\Gamma}(P_A) \setminus \overline{G}_1(p, P_A)$, then $D(q, P_A \cup \{p\}) = D(q, P_A)$ and $D(q, P_N \setminus \{p\}) = D(q, P_N)$.

ii) If $q \in \overline{G}_1(p, P_A)$, then $D(q, P_A \cup \{p\}) = D(q, P_A) + 1$ and $D(q, P_N \setminus \{p\}) = D(q, P_N) - 1$, and thus $\min\{k, D(q, P_A \cup \{p\}) + D(q, P_N \setminus \{p\})\} = \min\{k, D(q, P_A) + D(q, P_N)\}$.

■

▶ **Proposition 6.5.**

$$LB_1(P_A, P_R \cup \{p\}, P_N \setminus \{p\}) = \max\begin{cases} LB_1(P_A, P_R, P_N) \\ \max\limits_{q \in \overline{G}_1(p, P_A)} \{\eta(q, P_A, P_N \setminus \{p\})\} \end{cases}$$

◀

Proof. We prove that for every point q in $\overline{\Gamma}(P_A) \setminus \overline{G}_1(q, P_A)$, it holds that $\eta(q, P_A, P_N) = \eta(q, P_A, P_N \setminus \{p\})$, and therefore, only the points in $\overline{G}_1(q, P_A)$ need to be considered. The proof is similar to part i) of the proof of Proposition 6.4, except that only $D(q, P_N)$ and $D(q, P_N \setminus \{p\})$ are taken into account. If $q \in \overline{\Gamma}(P_A) \setminus \overline{G}_1(p, P_A)$, then $D(q, P_N \setminus \{p\}) = D(q, P_N)$. ■

Update of LB_2 For the second lower bound computation, we consider the following subset of $\Gamma(P_A \cup \{p\})$:

$$G_0(p, P_A) := \Gamma(P_A \cup \{p\}) \setminus \Gamma(P_A).$$

We state the following equalities.

▶ **Proposition 6.6.**

$$LB_2(P_A \cup \{p\}, P_R, P_N \setminus \{p\}) = \max\begin{cases} LB_2(P_A, P_R, P_N) \\ \max\limits_{q \in G_0(p,P_A)} \left\{ \frac{1}{k}\overline{D}(q, P_A \cup \{p\}) - \lambda(q) \right\} \end{cases}$$

◀

Proof. Similar to the proof of Proposition 6.4. If $q \in \Gamma(P_A)$, then we have that $\overline{D}(q, P_A \cup \{p\}) = \overline{D}(q, P_A)$ and $\overline{D}(q, P_N \setminus \{p\}) = \overline{D}(q, P_N)$. ■

The following proposition simply uses the fact that LB_2 is only defined via P_A. Moving a point from P_N to P_R does not have any effect on the value of this lower bound.

▶ **Proposition 6.7.** It holds that $LB_2(P_A, P_R \cup \{p\}, P_N \setminus \{p\}) = LB_2(P_A, P_R, P_N)$. ◀

The sets $\overline{G}_0(p, P_A)$ and $G_0(p, P_A)$ are of size $O(dk^{d-1})$ at worst, as we have d possible choices for a coordinate taken from p and, given this fixed coordinate, we have $(|P_A| + 1) \leq k$ choices for each of the other $d - 1$ coordinates. The set $\overline{G}_1(p, P_A)$ is of size $O(k^{d-1})$ as the first coordinate is fixed and we could have up to $|P_A| \leq k$ choices for each of the other coordinates in the worst case. The results above indicate that the lower bound can be computed incrementally in $O(dk^{d-1})$ time at each recursive step, assuming that $D(q, P_N)$ and $\overline{D}(q', P_N)$ can be computed in constant time after a pre-processing step as discussed in this section.

6.3.3 Greedy Heuristic

An initial upper bound for BB is given by a greedy heuristic that selects k points iteratively. The greedy choice consists of selecting the point amongst those that were not yet chosen that gives the best improvement in terms of star discrepancy (note here that this improvement can be negative, as discussed in Example 6.2). Therefore, the selection of the next point involves the evaluation of $O(n)$ star discrepancies, each of which takes $O(k^d)$ time with a naïve approach. Although

better running times can be achieved, we found this procedure to be reasonably fast for the size of the point sets considered in our experimental analysis. In the subsequent sections, we will include performance statistics for the greedy heuristic in our reports, to provide an impression for its quality in the various use-cases.

6.3.4 The Feasibility Approach

For practitioners interested only in good discrepancy values and not the optimal subset, a possible approach is to consider a *feasibility problem*. The idea is to use the same model as the MILP introduced in Section 6.3.1, but to remove the objective and replace it by a constraint with a target value d^*. In other words, we are directly solving the decision problem: rather than solving to find the optimal subset, we try to find if there exists a subset P_k such that $d^*_\infty(P_k) \leq d^*$. This formulation allows us to use solvers dedicated to *Constraint Programming* problems, to try to exploit the fact that we are only trying to check if the model is feasible or not.

We performed experiments with open-WBO [MML14] and observed that, while many instances could be solved nearly instantly, the closer d^* got to the correct optimal value found in our other experiments, the longer it took to solve the problem. In particular, it was easier for the solver to find a value $d^* > d_{OPT}$ close to the optimal d_{OPT} than one as close but for which $d^* < d_{OPT}$. Overall, while this method was efficient to find acceptable solutions, it did not perform noticeably better than the original MILP. It also suffered from the size of the model, and therefore did not allow us to increase n or d significantly compared to the MILP. Should there be major improvements in the Constraint Programming community, it could be interesting to revisit this approach.

6.3.5 A Simple Case: Dominated Points

Before moving on to the computational results, we conclude our study of exact algorithms for the L_∞ star discrepancy with a specific case that we can solve exactly in polynomial time.

The difficulty with our methods so far is the size of the grid. Even when looking exactly once at each box, we cannot do better than $O(n^d)$. Should P consist of dominated points (i.e $x^{(i)} < x^{(j)}$ if $i < j$), then there are only $(d+1)n$ relevant boxes. Each point defines a closed box $[0, x^{(i)}]$ and d open boxes of the shape $\left[0, (1, \ldots, 1, x^{(i)}_j, 1, \ldots, 1\right)($, and these are the only critical boxes for P. More importantly, if we consider our points from 1 to n, when deciding whether to take the i-th point we only need to know how many points we have chosen so far and not

which ones they are. Indeed, regardless of our choices for $1 \leq j \leq i-1$, the critical boxes defined by $x^{(i)}$ will contain the same number of points.

Let $V(i,j)$ be the best discrepancy value that can be obtained when picking in our subset j of the i first points. We have the following dynamic programming relation

$$V(i,j) = \min(V(i-1,j), \max(V(i-1,j-1), B(i,j))),$$

where $B(i,j)$ is the worst discrepancy value for the $d+1$ critical boxes defined by $x^{(i)}$ with j points picked previously.[37] This can be computed in $O(d)$ time:[38] since the points are sorted, we know the closed box will have j points and the open boxes $j-1$. Volumes are also known, respectively $\prod_{h=1}^{d} x_h^{(i)}$ for the closed box and $x_h^{(i)}$ for $h \in \{1,\ldots,d\}$ for the different open boxes. $V(n,k)$ can therefore be computed in time $O(nkd)$ and the usual dynamic programming back-tracking method gives us the subset with the smallest star discrepancy.

6.4 Comparison of the Different Algorithms

We have presented above three different strategies to address the discrepancy subset selection problem: an MILP formulation, the branch-and-bound algorithm, and the greedy strategy. In this section we compare the efficiency of these three algorithms. We add to the comparison a naïve random sampling approach, which simply selects random subsets of the target size k. Some of the tables are presented separately in Section 12.1 for completeness, while not overloading this chapter.

The MILP solver and the branch-and-bound algorithm do not always terminate within the given time limit. In these cases, they can nevertheless report the best solution that they have been able to find.

6.4.1 Experimental Setup

The Sobol, Halton and RevHal point sets were generated by a program written in C using GNU Scientific Library, namely, library gsl_qrng for the generation of quasi-random sequences, with the procedures gsl_qrng_sobol, gsl_qrng_halton, and gsl_qrng_reversehalton, respectively. The sequences unif were also generated in a similar way, using library gsl_rng for random number generation with the procedure gsl_rng_uniform. Faure and iLHS point sets were generated in R

37 This includes the point $x^{(i)}$.

38 Should we allow equal coordinates, this would no longer hold, and add in the worst case a $O(n)$ factor.

using procedure `runif.faure` available in the `DiceDesign` package and procedure `improvedLHS` available in the `lhs` package, respectively. `Fibon` sets were generated by a code in `Python` (version 2.7.16).

For the two-dimensional case, we considered $k \in \{20, 40, 60, 80, 100, 120\}$ and for each value of k, we considered $n \in \{k + 20, k + 40, \ldots, 140\}$. For the three dimensional case, we considered $k \in \{20, 40, 60, 80\}$ and for each value of k, $n \in \{k + 20, k + 40, \ldots, 100\}$. Preliminary experiments indicated that larger values of n would increase the computational cost significantly, requiring several hours of computation time before the algorithms converge. For the two randomized constructions `iLHS` and unif, we have generated 10 instances for each combination of values of k and n.

To compare the discrepancy values of the subsets with the original size-k point sets, we also computed the discrepancy values of the latter, using the (exact) algorithm described in [DEM96] and provided to us by Magnus Wahlström. For consistency, we denote these cases as "$n = k$".

We used SCIP solver version 7.0.1 to solve the MILP formulation described in Section 6.3.1. The MILP formulation was written in an LP format, which is read and solved by SCIP solver with the default parameters. The BB algorithm for two and three dimensions and with the incremental computation of lower bounds as described in Section 6.3.2 was written in `C`. In a preliminary step, the points were sorted in increasing order with respect to the first dimension to prepare them for the application of our solvers.

To run the experiments, we used a computer cluster Dell PowerEdge R740 Server with two Intel Xeon Silver 4210R 2.4G, 10 Cores / 20 Threads, 9.6GT/s, 13.75M cache, with two 32GB RDIMM, two 480GB SSD SATA hard-drives, and Debian GNU/Linux 10 (buster) operating system. The running times in seconds of the SCIP solver and of the BB program were measured with command `time` under linux, with a cut-off time limit of 30 minutes. The time to generate the files with the MILP formulation was not taken into account. For the BB program, we used gcc compiler version 8.3.0 with `-O3` compilation flag. We have only used arrays with static memory allocation.

6.4.2 Quality of Random Subset Sampling and the Greedy Heuristic

To gain a feeling for the complexity of the subset selection problem, we first study the solution quality of randomly selected subsets of target size k as well as that of

Table 6.1: Percentiles of the star discrepancy values found by random subset sampling with 1 000 000 trials, for the instances with $n = 100$ points and subset size $k = 60$ in dimension $d = 2$.

quantile	Faure	Sobol'	Halton	RevHal	Fibon
best possible subset	0.0357	0.0356	0.0359	0.0363	0.0351
best found subset	0.0547	0.0540	0.0531	0.0542	0.0518
1%	0.0718	0.0715	0.0714	0.0713	0.0688
10%	0.0844	0.0836	0.0843	0.0838	0.0808
25%	0.0937	0.0928	0.0942	0.0935	0.0899
50%	0.1065	0.1055	0.1078	0.1063	0.1025
75%	0.1219	0.1212	0.1245	0.1222	0.1177
90%	0.1382	0.1370	0.1418	0.1384	0.1334
100%	0.24481	0.2459	0.2673	0.2531	0.2478

the greedy heuristic described in Section 6.3.3 (i.e., the strategy used to initialize the upper bound for the BB method).

Table 6.1 shows selected percentiles of star discrepancy values for 1 000 000 i.i.d. uniformly selected subsets of size $k = 60$ for the five considered low-discrepancy sequences and sets with $n = 100$ points in dimension $d = 2$. The distributions are quite similar for the five point sets. The main probability mass is around about twice the solution quality of the best found subset. The latter, in turn, have discrepancy values that are still between 47.5% and 53.3% worse than the best possible subset. Even if the evaluation of 1 000 000 subsets could be executed in less than two minutes, the results already suggest that random sampling is quite inefficient for the discrepancy subset selection problem.

That the inefficiency of the random subset sampling is not an artifact of the setting described in Table 6.1 is indeed confirmed by the values in Tables 12.5 and 12.8 (available in Section 12.1), where we report, for all tested combinations of k and n in $2d$ and $3d$, respectively, the discrepancy values of the best random subset that could be found within a cut-off time of 30 minutes. For fixed k, the values do not significantly improve with increasing n, in contrast to the value of the best possible (or best found) subset of the same size, which are reported in column *subset*. As a result, the relative disadvantage of the random subset selection procedure increases from around 10% for $k = 20$ and $n = 40$ to around 40% for $n = 140$ in the $2d$ case. For $k = 40$, the disadvantage is already around 16% on average for $n = 60$ and 58% for $n = 140$. For $k = 120$ and $n = 140$, the relative disadvantage of random subset sampling is between 22% for Sobol and 32% for

Halton. For the $3d$ case, the best subsets found by random sampling are around 19% worse on average than the optimal ones for $k = 20$ and $n = 40$, and this value increases to around 30% for $k = 20$ and $n = 60$ and to around 35% for $k = 20$ and $n = 80$ and $n = 100$.

Comparing random subset sampling to the greedy strategy (column *greedy* in Tables 12.5 and 12.8 in Section 12.1), we see that random subset sampling provides a much better upper bound; however, we should keep in mind that several thousands of millions of subsets are evaluated within the 30 minutes time limit of the random subset sampling strategy, whereas the greedy strategy is deterministic and therefore evaluates only a single subset. As discussed above, the figures in Table 6.1 showed that already after two minutes the median performance of random subset sampling was around twice as large as the value of the best found subset, so that the comparison between random subset selection and the greedy strategy should indeed be done with care. For fixed $k \in \{20, 40, 60\}$, the greedy strategy tends to give worse solutions when the number of available points, n, increases. Across all evaluated settings, its discrepancy values are between 33% and 172% worse than the best (or best found) subset, with an average overhead of 92% and a median of 88%. The average and the median disadvantage of the greedy strategy compared to the result of the random subset sampling are both around 40%.

Most observations made for the low-discrepancy sequences carry over to the performance on the subset selection problem on iLHS and unif, as can be seen in Tables 12.6 and 12.7 for the $2d$ case and in Tables 12.9 and 12.10 for the $3d$ case, respectively, in Section 12.1. In particular, the performance of *random* subset sampling decreases with increasing n and fixed k, whereas the values of the best possible subsets improve. In fact, not only the relative but also the absolute value of the best subset found by random subset sampling increases with increasing n, and this consistently for all k in $2d$ in the case of iLHS point sets and for most values of k in the unif case (no correlation between different values of n can be identified for the case $k = 20$ nor in the two cases for the $3d$ setting). No clear correlation between the quality of the greedy strategy and the value of k and n can be identified, except that for the $2d$ iLHS samples the absolute values of the subset computed for $n = 140$ tend to be worse than those for smaller n. This effect, however, cannot be observed in the $2d$ unif nor in the $3d$ cases.

The bounds provided by the *greedy* strategy are quite stable for varying n and fixed k, but are significantly worse than bounds provided by the *random* strategy in $2d$. In $3d$, however, this is not the case. Here, the results of the *greedy* strategy are better than those of the random subset sampling; the average (median, max) advantage of the greedy strategy over the random one is 20% (30%, 38%) for iLHS and 16% (18%, 36%) for unif. In some of the $3d$ cases, the best value returned by

Table 6.2: CPU-time (in seconds) of BB for low-discrepancy sequences and sets and median CPU-time and number of instances solved out of ten (in parenthesis) for randomized constructions for several values of n and k in the three-dimensional case, where "-" indicates that the approach did not terminate before the time limit of 1800 seconds.

k	sequence	$n = 40$	$n = 60$	$n = 80$
20	Faure	34	859	-
	Sobol'	4	973	-
	Halton	30	-	-
	RevHal	10	1278	1378
	iLHS	161 (9)	620 (8)	567 (1)
	unif	31 (9)	305 (8)	966 (1)
40	Faure		-	-
	Sobol'		-	-
	Halton		-	-
	RevHal		-	-
	iLHS		253 (2)	-
	unif		-	-
60	Faure			-
	Sobol'			-
	Halton			-
	RevHal			-
	iLHS			806 (2)
	unif			86 (2)

the greedy strategy is either optimal (this is the case for the $k = 60$, $n = 80$, iLHS setting) or could not be improved by the exact solvers ($k = 80$, $n = 100$, iLHS; $k = 60$, $n = 80$, unif; and $k = 80$, $n = 100$, unif) or it is quite close to optimal (e.g., $k = 80$, $n = 100$, iLHS with a 0.5% overhead compared to the best value returned by the exact solvers). Note that the *random* strategy evaluates much fewer samples in $3d$ than in $2d$, since the star discrepancy computation is substantially more time-consuming in $3d$.

Thus, summarizing this section, we find that (with few exceptions), both the *random* subset sampling and the *greedy* heuristic perform rather poorly on the discrepancy subset selection problem, clearly motivating the need for more sophisticated approaches, either in terms of exact solvers such as the MILP and BB approaches presented in Section 6.3 or in terms of better heuristics.

6.4.3 Comparison between MILPs and Branch-and-Bound

Tables 12.3 and 12.4 in the appendix present the running times (measured in seconds) of the MILP solver and BB, for different values of n and k in dimension $d = 2$ on deterministic sequences and on randomized constructions, respectively. The same information for branch and bound in $d = 3$ is shown in Table 6.2 (for the randomized constructions more details can be found in Table 12.1 in Section 12.1). We do not show the results for the MILP solver, since they have shown poor performance in the $3d$ case. For the randomized sequences, the values reported in Table 6.2 are the median running times on the instances that were solved within the cut-off time. In Tables 12.1 and 12.4, we report the minimum, the median and the maximum value. The number of instances that were solved within the time limit is reported in parenthesis. The entry "-" indicates that the implementation was not able to terminate within this cut-off time.

For a better comprehension of the information shown in Tables 12.3 and 12.4, Figure 6.2 plots a summary of these tables. The left and the right column correspond to the performance of the ILP solver on the MILP formulation and of BB, respectively. Each row corresponds to the performance obtained for a given value of k. The points correspond to the running times in seconds obtained on deterministic sequences or sets (fau is Faure, sob is Sobol hal is Halton, rev is RevHal, and fib is Fibon) and to the median running times in seconds on randomized constructions (lhs is iLHS and uni is unif). The label for each deterministic sequence is placed close to the point with the largest value of n for which the approach was able to solve before the time limit of 1800 seconds was achieved. In the case of randomized constructions, the label is placed close to the point with the largest value of n for which the approach was able to solve at least one instance before the time limit. The barplots present the percentage of solved instances of randomized constructions, where violet and blue correspond to iLHS and unif, respectively.

The results for $d = 2$ in Tables 12.3 and 12.4 suggest two different patterns for MILP and for BB: while the latter is faster to find the optimal subset for small k/n ratios (see row $k = 20$ in Table 12.3 and second column and first row in Figure 6.2), MILP is faster for larger k/n ratios (see main diagonal in Table 12.3 and the leftmost running-times in the left column of Figure 6.2). The difference between the two methods is striking at both ends. BB solves all instances with $n = 140$ and $k = 20$ in almost less than 100 seconds whereas MILP cannot even solve a single one. MILP can solve all instances for $n = 140$ and $k = 120$, except for Fibon sets, whereas BB can only solve RevHal, Fibon, and almost half of the iLHS instances.

The strong performance of MILP as compared with BB when the k/n ratio is close to 1 is related to the quality of the lower bounds on those cases. Table 12.2

in the appendix shows the integrality gap of the LP relaxation of MILP for $2d$ deterministic sequences with respect to the optimal found and, when not available, with respect to the best solution found. We can observe that the smallest gaps arise for larger k/n ratios. We recall that the solution of the LP relaxation of MILP is integral when $k = n$.

Unfortunately, MILP is not feasible for the $3d$ case as the memory requirement grows very fast, reaching the limit available in the cluster for the smallest instances. This is due to the large number of constraints in the $3d$ case. In fact, a file with the MILP formulation in LP format occupies several gigabytes. For this reason, Tables 6.2 and 12.1 report only the CPU-time taken by BB on deterministic and randomized sequences, respectively. The performance decay of BB reported in these tables is noticeable in comparison with the $2d$ case. For instance, this approach cannot find a single solution for $3d$ instances with $n = 100$ and $k = 20$, whereas it can solve all $2d$ instances for the same values of k and n in at most 250 seconds. This is mainly due to the time taken with the update of data structures and the evaluation of the lower bounds, which grows considerably with an increasing number of dimensions.

We also observe from Table 12.3 that there seems to be little difference of performance among the different deterministic sequences and sets. Still, some outliers are quite noticeable with BB, such as with Faure sequence for $n = 100$ and $k = 80$ which took 1 194 seconds, while with Fibon sequence for the same parameters took only 45 seconds. Similarly, MILP was not able to solve a Fibon set for $n = 120$ and $k = 100$, took 1 538 seconds to solve a Sobol sequence with the same sizes, while it took less than 20 seconds to solve the remaining sequences. There is also no large difference between the running times obtained on deterministic sequences or sets and on iLHS sets. Differently, we observe that unif sets take less time to be solved with MILP while they take more time to be solved with BB.

Table 6.3 compares the final solution quality of the MILP and BB for $n = 140$ and different values of k, after a cut-off time of 23 hours. Values marked by an asterix * could not be proven to be optimal by the solver, and values printed in grey color are known to be non-optimal, by the result of the other solver. BB could solve all but four instances, whereas the solver for MILP did not finish on eleven instances. The solution quality, however, is nevertheless decent: only two values deviate from the best solution found by BB by more than 1%; these are for $k = 60$ and Sobol (1.8% worse than the optimal solution) and for Halton (2.1% worse). Only for the case of $k = 100$ for Sobol sequence (+2.1% compared to the optimal solution) and for Halton (+22.7% compared to the best solution found by MILP) are the values obtained by BB worse than those obtained by MILP.

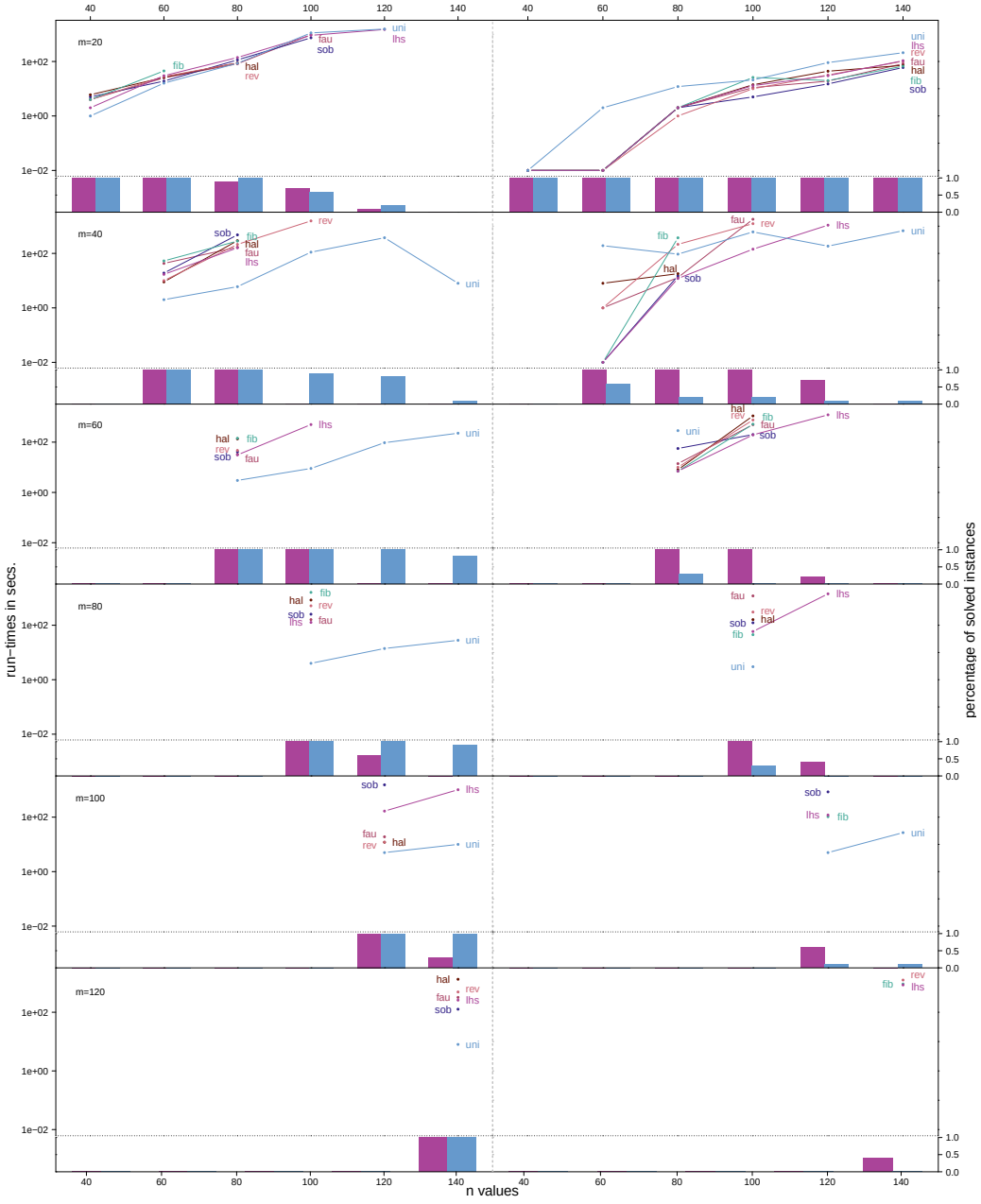

Figure 6.2: Run-times for deterministic sequences, median run-times for randomized sequences (points and lines), and percentage of solved instances for randomized sequences (barplots) for MILP (left column) and BB (right column) and for each combination of k (rows) and n.[39]

Table 6.3: Optimal and best found (*) star discrepancy values for MILP and BB with a time limit of 23 hours. All data is for the two-dimensional case with $n = 140$ and different values of k. Provably non-optimal values are printed in grey color.

k	sequence	$n = 140$	
		MILP	BB
40	Faure	*0.0449	0.0449
	Sobol'	*0.0449	0.0447
	Halton	*0.0452	0.0452
	RevHal	0.0444	0.0444
	Fibon	0.0448	0.0448
60	Faure	*0.0334	0.0334
	Sobol'	*0.0334	0.0328
	Halton	*0.0345	0.0338
	RevHal	*0.0336	0.0336
	Fibon	*0.0338	0.0338
80	Faure	*0.0273	0.0271
	Sobol'	*0.0273	0.0273
	Halton	0.0277	0.0277
	RevHal	*0.0279	*0.0278
	Fibon	*0.0296	*0.0276
100	Faure	0.0241	0.0241
	Sobol'	0.0241	*0.0246
	Halton	*0.0242	*0.0297
	RevHal	0.0238	*0.0238
	Fibon	*0.0296	0.0230

6.5 Comparison of Star Discrepancy Values

While we have focused in Section 6.4 on the comparison between the different solvers, we now discuss the quality of the subsets for the different point constructions. Detailed values and information about the convergence of the exact solvers can be found in Tables 12.5, 12.6, and 12.7 for low-discrepancy, iLHS, and unif samples in $2d$ and in Tables 12.8, 12.9, and 12.10 for low-discrepancy, iLHS, and unif samples in $3d$, respectively.

6.5.1 The Two-Dimensional Case

Figure 6.3 visualizes the star discrepancy values of the optimal (or best found, see Tables 12.5, 12.6, and 12.7 for details) subsets for all tested combinations of n and k in $2d$.

39 k is replaced by m in this plot and subsequent ones.

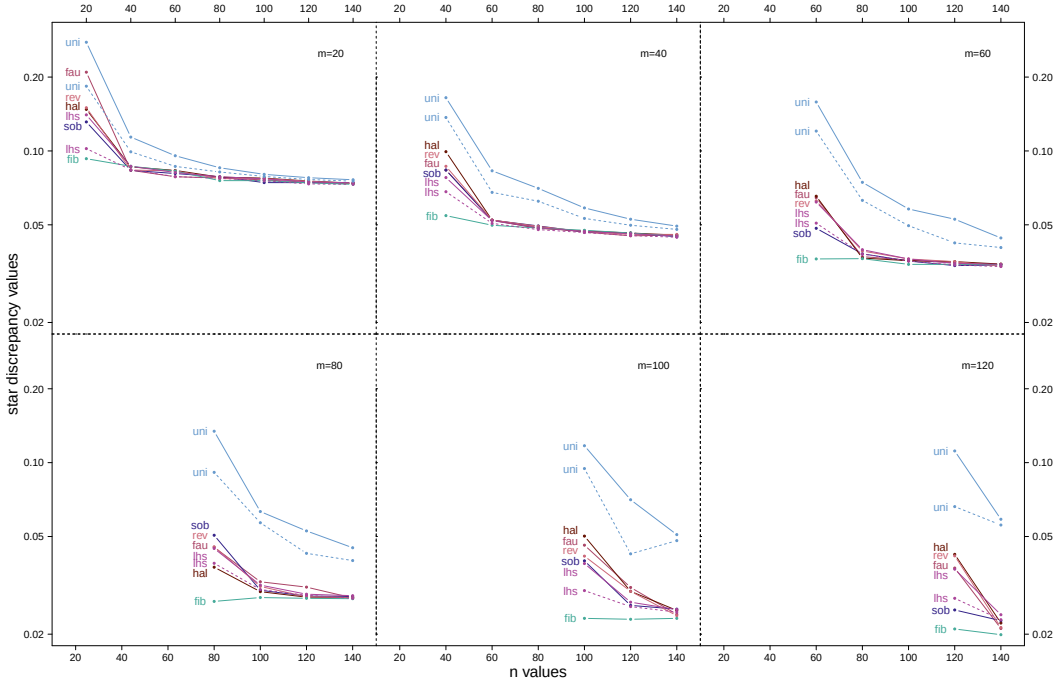

Figure 6.3: Star discrepancy values for each tested combination of k and n in $2d$. For the two randomized constructions `iLHS` and unif, minimum (dashed lines) and median (solid lines) values across the ten independent runs are shown.

Dependency on k. As expected, the discrepancy values decrease with increasing k. While the discrepancy of the best original construction with m points decreases from 0.093 to 0.0545, 0.0363, 0.0272, 0.0232, and 0.021 for $k = 20, 40, \ldots, 120$ points, the discrepancy value of the best found size-k subset (over all $n > k$ studied) decreases from 0.0731 for $k = 20$ to 0.0445 for $k = 40$, 0.0338 for $k = 60$, 0.0272 for $k = 80$, 0.023 for $k = 100$, and 0.0199 for $k = 120$. The advantage of the subset selection is therefore around 21% for $k = 20$, 18% for $k = 40$, 7% for $k = 60$, 0% for $k = 80$, 1% for $k = 100$, and 5% for $k = 120$.

Dependency on n. For fixed k, the values tend to decrease with increasing n, but there are a few cases that do not follow this rule, i.e., in which the optimal k point subset of a $n = k + 20i$ set has greater discrepancy value than the set with $n = k + 20(i - 1)$ points. Cases with $n = 140$ may be caused by non-convergence of the exact solvers, i.e., the reported bounds may simply not reflect the value of an optimal subset. Examples for this setting are Faure with $k = 40$, Sobol with $k = 60$, Fibon with $k = 100$. However, there are also cases in which the increase in discrepancy value is not caused by this artifact, but by a real disadvantage of

	sequence	m=20, best=0.0731 n=m	min	m=40, best=0.0445 n=m	min	m=60, best=0.0338 n=m	min	m=80, best=0.0272 n=m	min	m=100, best=0.023 n=m	min	m=120, best=0.0199 n=m	min
2d	Faure	186.5%	1.0%	87.9%	1.3%	90.8%	2.1%	66.2%	3.3%	100.4%	4.8%	86.9%	6.0%
	Sobol	79.6%	1.0%	87.9%	0.4%	43.2%	0.9%	86.0%	4.8%	73.0%	10.0%	26.1%	14.1%
	Halton	102.1%	1.1%	123.1%	2.0%	93.5%	2.4%	37.9%	3.7%	118.3%	8.7%	112.6%	11.6%
	RevHal	105.2%	0.7%	94.6%	2.7%	85.2%	0.3%	66.9%	3.7%	80.9%	3.5%	109.5%	7.0%
	Fibon	27.2%	0.0%	22.5%	0.9%	7.4%	1.8%	0.0%	0.0%	0.9%	0.0%	5.5%	0.0%
	iLHS (min/10 rep.)	39.8%	0.0%	53.3%	0.0%	50.3%	0.0%	43.0%	2.9%	30.9%	7.0%	40.7%	15.1%
	unif (min/10 rep.)	151.2%	3.0%	207.6%	7.6%	256.5%	19.5%	235.7%	46.7%	311.3%	96.5%	232.7%	179.9%

	sequence	m=20, best=0.1202 n=m	min	m=40, best=0.0778 n=m	min	m=60, best=0.0606 n=m	min	m=80, best=0.0547 n=m	min
3d	Faure	49.3%	0.2%	136.0%	0.0%	82.7%	0.0%	17.0%	16.6%
	Sobol	47.6%	0.0%	37.0%	1.0%	21.5%	6.1%	51.4%	10.6%
	Halton	73.0%	1.0%	89.6%	3.7%	78.4%	5.9%	28.0%	0.5%
	RevHal	55.6%	0.4%	71.3%	2.7%	42.9%	6.9%	36.6%	0.0%
	iLHS (min/10 rep.)	228.3%	7.3%	286.0%	25.6%	453.1%	74.3%	421.2%	212.4%
	unif (min/10 rep.)	240.2%	7.5%	333.8%	44.9%	457.4%	95.9%	483.2%	242.2%

Figure 6.4: Relative disadvantage of the discrepancy values of the original point sets (column "$m = n$") and of the best size-k subset (across all tested sets with $n = k + 20i$ points, column "min"), compared against the best overall set with k points. For the two random constructions, iLHS and unif, we report the best out of the ten independent experiments.

the larger n-point set. This is the case for the Fibon set with $k = 60$, where the discrepancy of the optimal subset of the $n = 80$ construction is 0.0364, slightly larger than the 0.0363 discrepancy of the original $k = 60$ construction. It is also the case for the Fibon set with $k = 80$, which has a discrepancy value of 0.0272 for the original ($n = k$) construction, whereas the optimal subset of the $n = 100$ point set has discrepancy 0.0282 (and also the best found subset for the $n = 120$ construction is worse than the original 80-point one, but the solvers did not converge, so that we do not know whether the disadvantage is real). Another example of a non-monotonic behavior is the unif construction with $k = 100$, but here the decrease in the discrepancy value of the best subset is simply caused by the random nature of the construction, and the comparatively large variance between the independently sampled n-point sets, see Figure 6.5 for an illustration.

We also observe a general trend for diminishing returns for increasing n, i.e., the relative gain when increasing n from k to $k + 20$ is larger than the gain when increasing n from $k + 20i$ to $k + 20(i + 1)$ for $i > 0$.

Comparison of the different constructions. The by far worst discrepancy values are obtained by the uniformly sampled point sets unif, and this even when considering the best of all ten independent runs (dashed line in Figure 6.3). For the original k-point constructions, i.e., the case $n = k$, the Fibon sets are clearly the best, with discrepancy values that are significantly smaller than that of all other constructions. However, we also see that the advantage of this set diminishes or even vanishes when considering the best size-k subsets that could be identified for $n > k$. Indeed, we observe that the discrepancy values of the $n = k$ point sets can differ quite

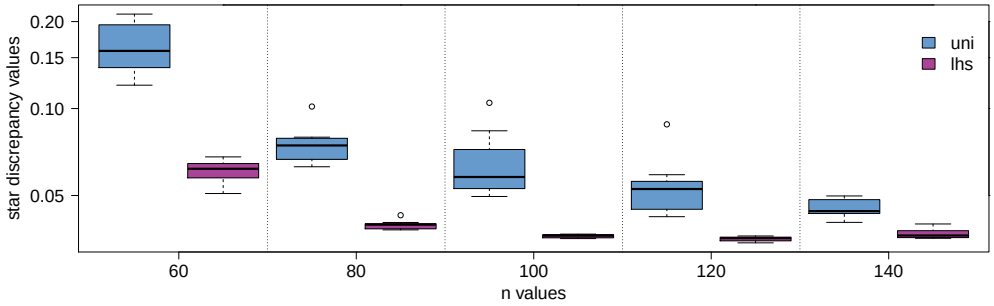

Figure 6.5: Boxplot of the star discrepancy values of the best sets found for iLHS and unif in $2d$ for $k = 60$ and for several values of n.

substantially between the different constructions, whereas their values are quite similar for the best (found) size-k subsets out of the $n = 140$ constructions.To analyze these values in more detail, we report in Figure 6.4 the smallest discrepancy value $d^*_\infty(P^*_k)$ found for any of the size-k point sets (top row, value reported as "best="). We then report in Figure 6.4 the relative disadvantage $(d^*_\infty(P) - d^*_\infty(P^*_k))/d^*_\infty(P^*_k)$ of the discrepancy value of the original k-point constructions (columns "$n = k$") and of the best found size-k subsets (column "min") against these best discrepancy values. The disadvantage of the original size-k constructions against the best found point set are quite significant for almost all constructions, with the exception of the Fibon set, for which the advantage of the subset selection approach varies only between 0% (for $k = 80$) and 27.2% (for $k = 20$). For all other constructions, we see a substantial advantage of the subset selection approach.

Taking the uniformly sampled point sets aside, the differences between the best size-k subsets are at most 1.1% for the case $k = 20$, at most 2.7% for the case $k = 40$, etc. These values increase with increasing k, but the plots in Figure 6.3 suggest that a further increase in n could reduce these differences. While the convergence itself may not be very surprising, it is interesting to see that a relatively small increase in n can suffice to find small discrepancy subsets in any of the low-discrepancy construction and in the iLHS sets. For uniformly sampled points, larger sample size n seems to be needed to achieve similarly small discrepancy values.

For the random point sets, the differences between the discrepancy values of the ten independent unif constructions are larger than those of the iLHS (sub-)sets, as can be easily seen from the examples plotted in Figure 6.5 and from the detailed values in Tables 12.6 and 12.7.

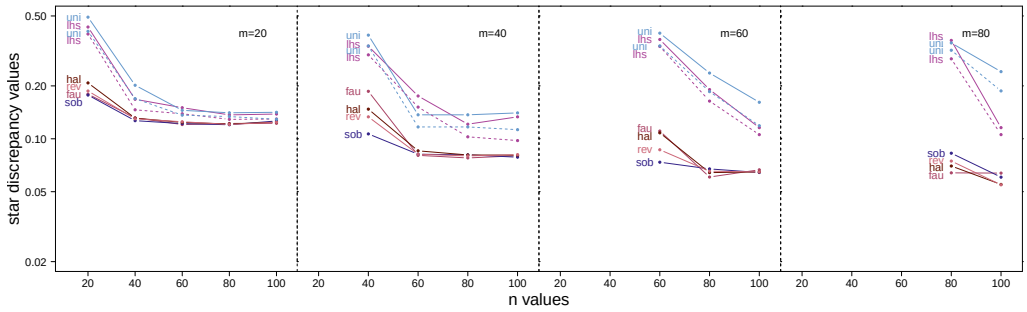

Figure 6.6: Star discrepancy values for each tested combination of k and n in $3d$. For the two randomized constructions `iLHS` and unif, minimum (dashed lines) and median (solid lines) values across the ten independent runs are shown.

6.5.2 The Three-Dimensional Case

Figure 6.6 compares the discrepancy values of the best (found) subsets of size k, for all tested super-sets of size n. Exact values of the best size-m point set and the relative disadvantages of the six considered constructions are provided on the bottom part of Figure 6.4, whereas detailed results are available in Tables 12.8 for the low-discrepancy sequences, 12.9 for `iLHS`, and 12.10 for unif, respectively.

Comparison with the $2d$ values. A construction that clearly stands out in the $2d$ case is the `fibon` set. This set, however, does not have a straightforward generalization to dimensions $d > 2$. It therefore does not appear in our $3d$ evaluations. Not surprisingly, the discrepancy values of the $3d$ constructions are much worse than that of the $2d$ constructions for any given k. For $k = 20$, the discrepancy of the best $3d$ set is 64% larger than that of the best $2d$ set of the same size. This disadvantage monotonically increases with k. It is 75% for $k = 40$, 79% for $k = 60$, and 101% for $k = 80$.

Dependency on k. As in the $2d$ case, the discrepancy values of the best found size-k point sets decrease with increasing k; they are 0.1202, 0.0778, 0.0606, and 0.0547 for $k = 20$, 40, 60, and 80, respectively. That is, the advantage of adding another 20 points decreases with increasing k. The discrepancy value of the best original ($n = k$) constructions are 0.1774, 0.1066, 0.0736, 0.064 for $k = 20$, 40, 60, and 80, respectively, resulting in a relative advantage of the best size-k subsets over the original ($n = k$) constructions decreasing from 32% to 27%, 18%, and 15%, respectively.

Dependency on n. For fixed k, the discrepancy decreases with increasing n, and this quite significantly already for $n = k + 20$, with an average gain of 44% in discrepancy value for $k = 20$ and $k = 40$, 36% for $k = 60$, and 26% for $k = 80$. The

latter values are based on incomplete data, however, since the algorithms did not converge in the 30 minutes time-out and therefore provided only upper bounds for the discrepancy values of the optimal subset. Based on the same data, the median gain in discrepancy values for $k = 20$, 40, 60, and 80 is 34%, 46%, 43%, and 27%, respectively. The values in Figure 6.4 and the curves in Figure 6.6 show that the advantage is slightly larger when considering the best subset across all tested n values, but – as in the $2d$ case – the advantage of increasing n from $k + 20i$ to $k + 20(i + 1)$ decreases rapidly for $i > 0$.

Comparison of the different constructions. Comparing the different sequences and sets, we observe a clear disadvantage of the unif and the iLHS constructions. Even the best found size-k (sub-)sets have a relative overhead of more than 7% for $k = 20$, more than 25% for $k = 40$, more than 74% for $k = 60$ and more than 200% for $k = 80$. The discrepancy values of the original k-point constructions (column $n = k$ in Figure 6.4) is above 200% for all settings, and it is even larger than 400% for $k = 60$ and $k = 80$. We recall that ten independently sampled constructions were evaluated, and the values reported in Figure 6.4 are for the best among these ten trials; the median and average would hence compare even more unfavorably (see Tables 12.9 and 12.10 for details).

The differences between the four low-discrepancy sequences are quite small for the best size-20 point set, with less than 1% difference. For $k = 40$, the largest difference between these sequences is 3.7%. For $k = 60$, the Faure sequence yields the best subset, and the best subsets of the other sequences are between 5.9% and 6.9% worse. For $k = 80$, the RevHal sequence has the best subset, closely followed by Halton. The best Sobol and Faure subsets are 10.6% and 16.6% worse. We suspect that the differences would decrease with increasing n, but we could not verify this assumption, since our algorithms did not converge for larger values of n.

Thus, overall, the low-discrepancy sequences show significant advantages over the two randomized constructions, but we do not see any clear ranking of these four tested sequences. For all point sets, the best size-k subsets have significantly smaller discrepancy than the original constructions with k points.

6.6 Conclusions and Future Work

We have introduced the star discrepancy subset selection problem and we have presented two different exact solvers, one based on mixed-integer linear programming (MILP) and one based on branch and bound (BB). We compared the performances of these solvers, and contrasted them with that of random subset sampling and a greedy construction. For the two-dimensional case, while the MILP solver is efficient

for large k/n ratios, BB seems more suitable for small k/n ratios. We relate these findings with the quality of the lower bounds. However, for the three-dimensional case only BB is able to solve this problem, even for small n.

Comparing the optimal subsets of seven different point constructions, our key findings are that (1) the discrepancy of the best size-k subset can be significantly better than the original size-k construction (with the only exception of the Fibon set with $k \geq 80$) and the main improvement stems from increasing n from k to $k + 20$, (2) the values of the best found subsets are very similar for all low-discrepancy constructions, regardless of their comparatively large differences in the original $k = n$ constructions; (3) unif and iLHS point sets are not competitive in $3d$ in terms of discrepancy values.

Given that many computer science applications operate with a fixed budget problem dimension d and a fixed budget n of points that can be evaluated, we consider it valuable to collect point sets of small discrepancy values. Our work shows that the subset selection approach could be an interesting alternative to construct such point sets. In the current setting, dimension 2 and relatively few points, it would be interesting to compare our results to other existing constructions, such as the symmetrized Fibonacci sequences suggested in [BTY12] or generalized Halton sequences [BW79]. Our sets could also serve both as an objective for future optimization-based approaches, and as an extra tool to further improve sets obtained via other methods.

The most natural next step remains to find methods of solving this problem in higher dimensions and with more points. Given that our initial approach was driven by low-discrepancy points sets having theoretically poorer performance in higher dimensions, we expect even better results in that context. The next chapter gives an example of such a heuristic approach.

6.7 The L_2 Version of Subset Selection

We finish this chapter with a short description of the L_2 version of this problem. The results presented here were not published but recent progress in the construction of good L_∞ point sets by using the L_2 discrepancy leads us to believe this could be useful in the future. As mentioned in Section 4.1, the main advantage of the L_2 discrepancy is the Warnock formula, providing both simplicity and a fast calculation. For our purposes, as for the Kritzinger sequence in Chapter 8, it also has the advantage of clearly describing the contribution of each point. While one could link this to

graph problems,[40] we will present here a reformulation as an *Unconstrained Binary Quadratic Problem (UBQP)*. An UBQP is formulated as

$$\text{minimize } y^T Q y$$

where $y \in \{0,1\}^n$ and Q is an n x n symmetric matrix.

There are two steps in the reformulation. Firstly, since we are working with a fixed set of points in a fixed dimension, we can ignore the $1/3^d$ constant term in equation (4.1). The first sum is naturally connected to terms of the shape $y_i Q_{i,i} y_i$, whereas the double sum will need to be split between the different coefficients. For the $Q_{i,i}$ coefficients, we have

$$Q_{i,i} := \frac{2^{1-d}}{n} \prod_{k=1}^{d} (1 - (x_k^{(i)})^2) + \frac{1}{n} \prod_{k=1}^{d} (1 - x_k^{(i)}),$$

by taking the i-th term from the first sum in equation (4.1) and the $i = j$ term in the second. For the other $Q_{i,j}$ coefficients, we have

$$\frac{2}{n^2} \prod_{k=1}^{d} (1 - \max(x_k^{(i)}, x_k^{(j)}))$$

by considering the two terms in the second sum.

When choosing y as the binary vector such that $y_i = 1$ if we pick the i-th point in our subset, we obtain the desired representation of our model, up to the constraint that we must choose exactly k points. As described in [Koc+14], this can be included in the objective function with a large enough penalty (i.e. any solution that respects the constraint would have a smaller solution value than any that does not). A constraint of the type $Ay = b$ can be replaced by a penalty of the shape $P(Ay - b)^T(Ay - b)$, with P the penalty value. In our case, since we are calculating the discrepancy value of a point set, between 0 and 1, and since $Ay = b$ can take only integral values, $P = 1$ suffices. The $P(Ay - b)^T(Ay - b)$ penalty can itself be rewritten as $y^T D y + c$ where c is a constant. In our case, we want to express the constraint that $\sum_{i=1}^{n} y_i = k$. This corresponds to the matrix with the first row equal to 1 and the rest 0, while $b_1 = k$ and 0 otherwise.

40 The contribution of a point in the sum with quadratic terms becomes a vertex weight, while the double sum is associated to edge weights on a complete graph.

Overall, this gives us the following problem to solve:

$$\text{minimize } y^T Q y,$$

where $Q = (q_{i,j})_{1 \leq i,j \leq n}$ is given by

$$q_{i,j} = 1 + \frac{1}{n^2} \prod_{k=1}^{d} (1 - \max(x_k^{(i)}, x_k^{(j)}))$$

if $i \neq j$ and

$$1 - 2k - \frac{2^{1-d}}{n} \prod_{k=1}^{d} (1 - (x_k^{(i)})^2) + \frac{1}{n^2} \prod_{k=1}^{d} (1 - x_k^{(i)})$$

otherwise.

The survey [Koc+14] gives a review of methods to solve this type of problem. While exact algorithms exist for a few points, it looks like heuristics such as tabu search would be needed to tackle this problem when we have a few thousands of points. The most interesting characteristic of this reformulation is that once the matrix Q has been created in $O(dn^2)$ time, solving the problem only depends on n and not d. This suggests it could be possible to tackle problems in very high dimensions, as long as n is in the thousands. We conclude this brief description with two remaining questions. The first is the complexity class of the problem. While subset selection for the L_∞ star discrepancy is unsurprisingly NP-hard, like the L_∞ discrepancy calculation, we were not able to find a reduction from a known problem to this one. The main reason is that there is a very precise structure in the matrix coefficients, and fixing a few of them leads to choosing many of them.[41] The second remaining question, arguably the most important, is how does this perform in practice. Can it provide sets with better L_∞ star discrepancy, by reaching much bigger n than the L_∞ version of the problem?

41 UBQP in general is NP-hard, and we could also formulate this as an NP-hard clique problem. The question is whether this always links to easy versions of these problems, or if we can at least once find a difficult version.

7 Subset Selection: a Heuristic Algorithm

This chapter corresponds to [CDP24], and is joint work with Carola Doerr and Luís Paquete.

7.1 Summary of Results

As the exact methods introduced in the previous section for the L_∞ star discrepancy present clear limits when n and d increase, we provide in this chapter a heuristic to solve the problem in much higher dimensions, as well as with a higher number of points. We introduce a swap-based heuristic, which attempts to replace a point of the chosen subset by one currently not chosen, using the box with the worst local discrepancy to guide our swap-choice. A further brute-force check is then used to guarantee that the final point set is a local minimum. This approach is able to obtain point sets of much smaller discrepancy than the known constructions for all dimensions for which the discrepancy can be reliably computed, hence significantly improving on the range of settings that can be handled by the exact methods presented in Chapter 6.

With different instantiations of the heuristic, we obtain point sets that are between 10 and 40% better than the initial Sobol' set of the same size. As for the exact methods, initial experiments show that choosing a subset of size k close to the initial point set size n leads to better results, especially when the dimension d and the target set size n increase, which was also observed with exact methods.

We also compare our method with the energy functional introduced by Steinerberger [Ste19] (see Section 4.2 for the one-dimensional version), which is minimized by gradient descent to obtain a point set with low discrepancy. His approach can be used on any point set in any dimension but he provides results mostly for dimensions 2 and 3. We give a detailed comparison of his method with ours as well as some extended testing in higher dimensions in Section 7.3.4. We show that not only can we clearly outperform the results obtained by this approach, but also that combining the two methods allows us to build point sets whose discrepancy is competitive with that of the Sobol' sequence. This can be done with any starting point set, it does not require a good number-theoretic construction. We note that the sets

obtained this way are not as good as those obtained by using subset selection on the Sobol' sequence.

Finally, our experiments provide numerous discrepancy values for the Sobol' sequence for varying n and d. It is conjectured in [NW10] that $n = 10d$ points are required to obtain a discrepancy of 0.25 in dimension d. Our results in Section 7.3.3 show that the Sobol' sequence seems to come close to a discrepancy of 0.2 with $n = 10d$ points in all tested dimensions. Furthermore, our improved sets obtained via subset selection come closer to $n = 7d$ points required to reach a discrepancy of 0.25.

Section 7.2 introduces the new heuristics to solve the problem in higher dimensions. Section 7.3 describes our numerical evaluation and the quality of the obtained point sets, as well as a comparison with Steinerberger's energy functional. Extensive numerical results and an extra proof are in Section 12.2 and Section 7.5 respectively. Our code and obtained point sets are available at https://github.com/frclement/SDSSP_Heuristics.

7.2 A Heuristic Approach for the Star Discrepancy Subset Selection Problem

To generalize the results of SDSSP to higher dimensions and to larger point sets, we introduce in this section a general method and several instantiations of it. The main working principle of this method is to keep the best subset found so far and, at each step, to attempt replacing some of the points inside this subset with some of the currently not chosen ones. The point set with the best discrepancy is then kept: either the initial one or the set obtained after the swaps. In case of a tie, the initial set is kept. For $j \in \{1, \dots, n-1\}$, we call *j-swap* the simultaneous replacement of j points inside the set with j points outside the set.

The main idea is to keep a current subset and to improve it via well-chosen 1-swaps. During a first step, we only consider the points defining the worst local discrepancy box as candidates to be removed from the chosen subset. That is, if the discrepancy of the current subset P^* is attained in $q \in [0,1]^d$, up to d points $x \in P^* \cap [0,q]$ with $x_i = q_i$ for some $i \in \{1, \dots, d\}$ are considered to be replaced by a point $y \in P \setminus P^*$. If no improving 1-swap is found, we then consider all remaining 1-swaps during the second step. At any point, if an improving 1-swap is found, we go back to the beginning of the first step with our new point set.

We note that our current choices for the heuristic are strongly influenced by the cost of discrepancy calculations. A single run with the brute-force check can require thousands of discrepancy calculations, each with a cost of $O(k^{d/2+1})$ for the

Algorithm 2: Pseudocode of our heuristic subset selection strategy with nb_{iter} restarts

Input: P, d, n, k.
Let $c[0, ..., d-1]$ be the critical box table
Let π_j be an ordering of $P \cup \{1, \ldots, 1\}$ in dimension j, for all $j \in \{0, \ldots, d-1\}$.
$d_{min} = \infty$, $P_{min} = \emptyset$
for $it = 0$ to $nb_{iter} - 1$ **do**
  Select P_k randomly from P, $|P_k| = k$
  while P_k is not a local minimum **do**
    found=False
    for $i = 0$ to $n - 1 - \min(c[0], \ldots, c[d-1])$ **do**
      for $j = 0$ to $d - 1$ **do**
        if $\pi_j(c[j]) + i < n$ **then**
          if $d^*_\infty(P_k) > d^*_\infty(P_k \backslash \{x^{(\pi_j(c[j]))}\} \cup \{x^{(\pi_j(c[j]+i))}\})$ **then**
            $P_k = P_k \backslash \{x^{(\pi_j(c[j]))}\} \cup \{x^{(\pi_j(c[j]+i))}\}$
            Update $c[0, \ldots, d-1]$, found=True
            break
      if found=True **then**
        break
    if found=False **then**
      Try all remaining swaps until one improves the set or all have been tested (P_k is a local minimum).
  if $d^*_\infty(P_k) < d_{min}$ **then**
    $d_{min} = d^*_\infty(P_k)$
    $P_{min} = P_k$
Return(P_{min}, d_{min})

DEM algorithm. Choosing carefully which swap to try should be a key focus in designing heuristics to tackle this problem. For ease of explanation, we will only consider the case when a closed box reaches the maximal local discrepancy value, the open box case is treated very similarly. A slightly simplified pseudo-code is provided in Algorithm 2.

In our experiments, to compare the performance of both discrepancy calculation methods in the context of subset selection, we will highlight which of the DEM algorithm and TA heuristic are used in the experiments.

Initialization. Let $P \subseteq [0, 1]^d$ with $|P| = n$ be the input point set and $k \leq n$ the target size. For each coordinate $i \in \{1, \ldots, d\}$, let π_i be the permutation ordering

the points $P \cup (1, \ldots, 1)$ by their i-th coordinate. In other words, $x_i^{(\pi_i(1))} \leq x_i^{(\pi_i(2))} \leq \ldots \leq x_i^{(\pi_i(n+1))} = 1$.

The algorithm is initialized with a randomly selected subset P_1, for which the discrepancy $d_\infty^*(P_1)$ and the corner $a = (a_1, \ldots, a_d)$ of the closed box $B_1 = [0, a]$ defining this discrepancy value are computed. To improve this subset, points in $B_1 \cap P_1$ must be replaced by points in $P \backslash (B_1 \cap P_1)$, otherwise the local discrepancy for B_1 will at best stay the same. Since the maximum local discrepancy can only be reached for a critical box, in every dimension $j \in \{1, \ldots, d\}$ there exists $x^{(\pi_j(c_j))}$ with $c_j \in \{1, \ldots, n+1\}$ such that $x_j^{(\pi_j(c_j))} = a_j$ and $x^{(\pi_j(c_j))} \in B_1$. These points will be called *edge points*, written in the table $c[0, \ldots, d-1]$ in Algorithm 2.

Step 1: Breadth-first search. A dimension $j \in \{1, \ldots, d\}$ is picked at random and the heuristic checks if $x^{(\pi_j(c_j+1))}$ is in P_1. If it is not, we compute the discrepancy of $(P_1 \backslash \{x^{(\pi_j(c_j))}\}) \cup \{x^{(\pi_j(c_j+1))}\}$. If this discrepancy is *strictly* lower than that of P_1, the chosen subset is changed to $P_2 := (P_1 \backslash \{x^{(\pi_j(c_j))}\}) \cup \{x^{(\pi_j(c_j+1))}\}$. There is a new critical box B_2 returned during the discrepancy computation and we go back to the beginning of Step 1. If $d_\infty^*(P_2) \geq d_\infty^*(P_1)$ or if $x^{(\pi_j(c_j+1))}$ is in P_1, dimension $j+1 \mod d$ is then considered, where we do the same operations. This continues until either we have found a better subset or all dimensions have been considered. If all dimensions are checked without finding an improvement, the heuristic goes back to the first dimension considered. It then does the same operations, but with $x^{(\pi_j(c_j+2))}$ rather than $x^{(\pi_j(c_j+1))}$ for all $j \in \{1, \ldots, d\}$. This continues until either we find a new subset with better discrepancy or until we have tried all possible swaps between $x^{(\pi_j(c_j))}$ and $x^{(\pi_j(b_j))}$, with $j \in \{1, \ldots, d\}$, $b_j \in \{c_j + 1, \ldots, n+1\}$, and $x^{(\pi_j(b_j))} \notin P_1$. In the first case, we go back to the beginning of Step 1 with our new point subset and new worst box.

Step 2: Brute-force check. If none of the swaps were successful, the heuristic tries all remaining valid swaps until either a better subset is found (in which case we go back to Step 1) or until we can guarantee that our current point set is a local minimum. The valid swaps to check are of two types. They can either involve an edge point: for any $j \in \{1, \ldots, d\}$, $x^{(\pi_j(c_j))}$ is swapped with $x^{(\pi_j(b_j))}$ for any b_j such that $b_j \in \{1, \ldots, c_j - 1\}$ and $x^{(\pi_j(b_j))} \notin P_1$. Or they swap a point strictly inside the box with one outside: $x^{(\pi_j(a_j))}$ such that $a_j \in \{1, \ldots, c_j - 1\}$ and $x^{(\pi_j(a_j))} \in P_1$ is swapped with $x^{(h)} \notin P_1 \cup B_1$ with $h \in \{1, \ldots, n\}$.

Restarts. Multiple runs of the heuristic with different starting positions are performed, to limit the influence of the initial subset.

7.2.1 Variants of the Algorithm

Rather than the current breadth-first search, we also tried a *depth-first* search where all swaps involving $x^{(\pi_j(c_j))}$ for a given j were done before moving to the next dimension. This did not give any noticeable improvements, and we expect our current method to perform better as it should swap points that are on average closer, with less risk of unbalancing our subset. Efficiently finding the optimal swap and the correct number of swaps to perform at once are both open questions.

Another possible modification of our algorithm would be to allow changes in the current set if the new discrepancy is equal to, and not only strictly smaller, than the current one ("*plateau moves*"). However, this less demanding selection strategy results in much worse empirical performance for our settings. We attribute this performance loss to the fact that one can keep the same discrepancy value while breaking the structure in other areas of the point set (for example, moving two points inside the worst box). This leads to the algorithm making the general structure of the point set worse while not changing the overall discrepancy value but blocking future improvements. These issues could possibly be mitigated by considering other information for tie-breaking, but we did not experiment with such ideas.

Non-guaranteed Optimality of Our Heuristic: While Section 7.3 will show the algorithm's promising performance, we note that we cannot have any theoretical guarantee for the optimality of the returned set. As the proposition in Section 7.5 shows, it is possible to return a local optimum that is not a global optimum. While the example shown in the proof is quite specific, such examples appear even for low n in dimension 2 in our experiments. An example is provided in Figure 7.1.

7.3 Experimental Study

In this section, we study the performance of our heuristic in different dimensions and slightly different instantiations. We also consider how the performance evolves when two of the problem's parameters are fixed, for example d and k or d and n. All experiments are done on the Sobol' sequence as it gave the best discrepancy values before using subset selection. Our methods nevertheless work on any sequence or set, they only depend on the quality of the initial set to provide good results (hence our choice of Sobol'). Indeed, subset selection on random sets provides sets with much better discrepancy than the initial random set, but not as good as known low-discrepancy sets and sequences (see Section 7.3.4).

We also use our discrepancy calculations to provide more empirical evidence

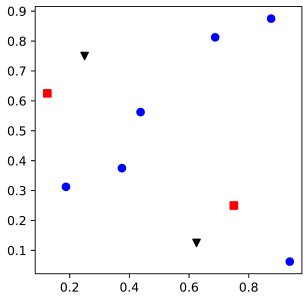

Figure 7.1: Two subsets for $k = 8$ taken from the first $n = 10$ points of the Sobol' sequence in dimension 2. The ten initial points are shown, with those present in both subsets shown as blue circles. The points shown as red squares and the blue points form a local optimum for 1-swap with discrepancy 0.234. The black triangles plus the blue points correspond to the global optimum of discrepancy 0.203. Neither of the two sets can be improved via 1-swaps. Intuitively, replacing the lower red square by the lower black triangle would create an overfilled box at the bottom, whereas replacing it by the upper triangle would create an overfilled box on the left.

regarding conjectures on the *inverse star discrepancy*, i.e., the number of points required in a given dimension to obtain a set of discrepancy less than a given threshold. We show in Section 7.3.3 that $n = 10d$ points seem to be sufficient to reach a discrepancy of 0.2. We conjecture that $n = 7d$ is a closer estimate to the inverse star discrepancy of 0.25. Finally, we provide in Section 7.3.4 a detailed comparison with Steinerberger's energy functional and a potential application of the combination of the two methods.

7.3.1 Experimental Setup

All the different parts of the code were done in C. The Sobol' sequence generation was done with the GNU Scientific Library, using the procedure gsl_qrng_sobol. Whenever randomness was required (for the first chosen subset and the dimension choice in Step 1 of the heuristics), `rand()` was used and initialized with `srand(1)`. The heuristics were implemented by us, with the exception of the two methods for calculating the star discrepancy (DEM algorithm and TA heuristic) which were provided to us by Magnus Wahlström and are available at [Clé+23b].

The experiments were run on a Debian/GNU Linux 11 computer, with a Quad Core Intel Core i7-6700 processor and 32 GB RAM. gcc 10.2.1 was used with the `-O3` compilation flag. Experiments were run with four types of heuristic instantiations,

either with the TA heuristic or the DEM algorithm, each with or without the brute-force check. The instantiations will be referred to as TA_BF, TA_NBF, DEM_BF or DEM_NBF, starting with TA if it is the TA heuristic and ending with BF if it did the brute-force check, NBF otherwise. Discrepancy values from DEM instantiations are exact but those from the TA ones are only lower bounds. They should nevertheless be relatively reliable below dimension 10.

We considered points in dimensions $\{4, 5, 6, 8, 10, 15, 25\}$. Unless specified otherwise, all ground sets are taken from the Sobol' sequence. Initial experiments were run for $n \in \{100, 150, 200, 250\}$ and $k \in \{n-10, n-20, n-30, n-40, n-50, n-60, n-70\}$, with a maximum of 10 runs for each instance. Further experiments to refine the parameter choices or get more precise results were done with slightly different values of n and k (but still of the same order of magnitude). They will be described for the relevant results. Each heuristic experiment was given a 1 hour cutoff, with the best value found so far returned if the heuristic had not finished by then. More precisely, if all 10 runs could finish in one hour then the value returned is the best of those, but for the larger instances the value returned may be the best value found in the first unfinished run.

7.3.2 Experiment Results

We describe here our experimental results, from general tests to have a global view on the performance of the heuristic, to finding the optimal parameters. The end of the section includes a brief discussion on comparisons with both sets and random subset selection.

General tests: Figures 7.2 and 7.3 show the performance of our different instantiations in dimensions 3 and 6, respectively. Plots change color to highlight when we are changing the ground set from which these points are selected (i.e., when we increment the previous n by 50). Whenever a heuristic was not able to terminate, we plot the best discrepancy value obtained during the run.

Figure 7.2 shows the performance of the different instantiations compared to the Sobol' sequence (black) in dimension 3. DEM_BF is the best performing version of the heuristic, with DEM_NBF outperforming it only for the two smallest instances. For $k \geq 70$, DEM_BF improves over the Sobol' set of the same size by between 17 and 27%. DEM_NBF is also performing better with a 14% decrease in the discrepancy on average. However, the TA variants are struggling: TA_NBF improves the Sobol' sequence by only 8% on average and TA_BF improves it by 2%, both being worse than the Sobol' set of similar size for numerous instances. In dimension 3, the DEM algorithm is much faster than the TA heuristic which has a similar runtime regardless of the dimension. This allows the DEM instantiations to run multiple

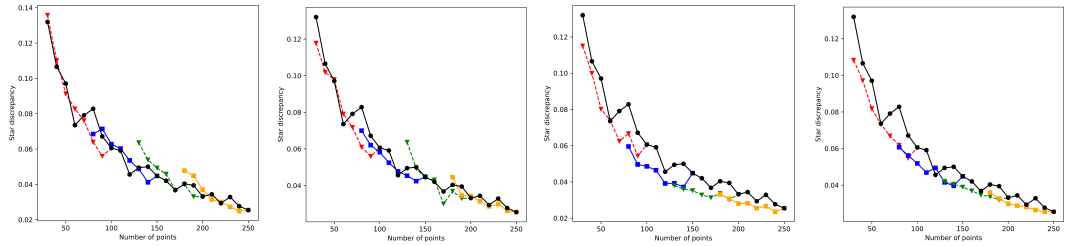

Figure 7.2: Performance of the different instantiations in dimension 3, from left to right: TA_BF, TA_NBF, DEM_BF and DEM_NBF. Different colors indicate a change of the initial set size (red for $n = 100$, blue for $n = 150$, green for $n = 200$ and yellow for $n = 250$), and the black curve corresponds to the Sobol' sequence (it is the same in all four plots). The plot includes the $k = n$ case for all four different n, the rightmost point in this color.

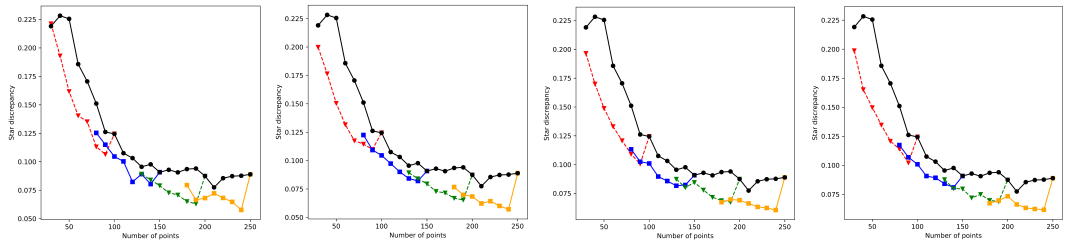

Figure 7.3: Performance of the different instantiations in dimension 6, from left to right: TA_BF, TA_NBF, DEM_BF and DEM_NBF. Different colors indicate a change of the initial set size (red for $n = 100$, blue for $n = 150$, green for $n = 200$ and yellow for $n = 250$), and the black curve corresponds to the Sobol' sequence (it is the same in all four plots). The plot includes the $k = n$ case for all four different n, the rightmost point in each color.

instances within the 1 hour time limit. For example, TA_BF does not finish even once for $n = 200$ and $k = 130$, whereas DEM_BF takes around 41 seconds for a single run. For $n = 150$ and $k = 80$, DEM_BF finishes a run in 6 seconds, DEM_NBF in 0.8 seconds and TA_NBF in 500 seconds.

Figure 7.3 shows that all 4 instantiations have relatively similar performances in dimension 6, all 4 having the best performance on at least one instance. On each instance, the best performing heuristic gives a 10 to 35% improvement on the discrepancy of the Sobol' sequence of the same size. Even taking the worst performing one, with the exception of 30 points for the TA_BF heuristic, we have a 7 to 30% improvement. We note that from $n \geq 120$ and $n - k \geq 30$ onwards, both _BF instantiations are often (or always for DEM_BF) unable to finish a single run. We introduced the subset selection problem largely because in higher dimensions a smaller number of samples might not guarantee that low-discrepancy sets would

have enough points to reach the asymptotic regime. The tests in dimensions 3 and 6 show that subset selection is more effective in dimension 6, despite the instantiations running fewer tries. For example, DEM_BF does not finish a single run in dimension 6 but it does 10 separate runs in dimension 3. This seems to confirm our hypothesis that subset selection will perform better in higher dimensions, for which an exponential number of points is required to reach the asymptotic bounds.

Fixed n or k: Given these initial results, further tests were done only on the DEM_BF and TA_NBF instantiations. In lower dimensions, the DEM algorithm is faster than the TA heuristic. Each heuristic run is fast enough to allow for a brute-force check and the DEM_BF instantiation gives us the best possible subsets with our method. It also guarantees the correctness of the discrepancy value. For higher dimensions, both the DEM algorithm and the brute-force check become too expensive, TA_NBF is the only reliably fast instantiation.

We now fix the dimension (here $d = 6$) and the resulting point set size ($k = 90$) to find the best ground set size n to obtain higher quality point sets. In Figure 7.4, we see that DEM_BF performs well for all different values of n, whereas TA_NBF works better for $n - k$ close to 20. The first 90 points of the Sobol' sequence have discrepancy 0.126, and only a single instance for TA_NBF fails to improve this value. The best instances, for DEM_BF with $n - k \in \{10, 20, 30\}$, give a 20% improvement over the Sobol' sequence. These values for $n - k$ are linked to our results on exact methods in Chapter 6, where the greatest discrepancy improvements were also observed for $n - k = 20$.

For $n_1 > n_2$ and a fixed k, the optimal subset selection solution for n_1 is better than the one for n_2. However, we observe that increasing n is not necessarily a good strategy for our heuristic, in particular for TA variants. We expect this to come from a larger search space which cannot be well explored by the heuristic and from the existence of a large number of sub-optimal local optima.

We then performed a similar experiment with fixed n and varying k to verify that our results were coherent, this time in dimension 5. The results are shown in Figure 7.5. We first note that the increase in discrepancy when k decreases is expected as we have smaller point sets. The heuristic performs well for all different values of k, with both instantiations always outperforming the Sobol' sequence. DEM_BF improves the Sobol' sequence discrepancy by between 12 and 33% and TA_NBF by 2 to 26%. Once again, TA_NBF performs much better when the difference between n and k is small whereas DEM_BF seems to be much more reliable for all values. The discrepancy of the point sets obtained with the heuristics behaves less erratically than the initial Sobol' sequence as the new point sets avoid discrepancy spikes for specific point set sizes. The noticeable improvement obtained by going from 200 to 195 points suggests that removing very few carefully

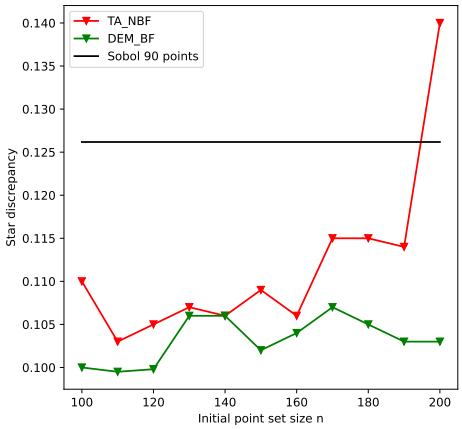

Figure 7.4: Best discrepancy obtained for different values of n, k fixed to 90, and $d = 6$, with a cutoff time of 1 hour.

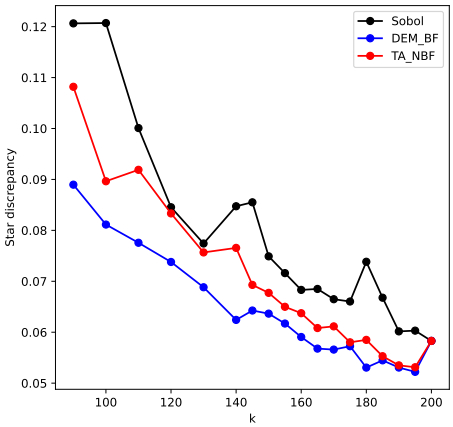

Figure 7.5: Discrepancy obtained for different values of k, n fixed to 200 and d=5.

chosen points could also lead to large discrepancy improvements. While we expect this to happen because our heuristics perform better (the search space is much smaller), this could be a promising direction for cheaper methods of improving low-discrepancy point sets.

Best results: Figure 7.6 gives the best values obtained by TA_NBF and DEM_BF for $k = n - 20$ and $k = n - 30$, which should be the best conditions for our heuristic given the previous results. We notice that there is very little difference between both algorithms, and with $k = n - 20$ or $k = n - 30$. In all cases, our heuristic is clearly outperforming the Sobol' sequence, the discrepancy value for 170 points of Sobol' is reached at 120 or 130 points for all our point sets. Our heuristic improves the Sobol' sequence's discrepancy by 8 to 30% depending on the instantiation choice. For each choice of k and n, the worst instantation improves the discrepancy by between 8 and 25%, whereas the best performing one improves it by 15 to 30%. While the plots here show results in dimension 6, our experiments show that our heuristic performs well for all dimensions for which we can compute the discrepancy. We note that results become poorer for much larger n: if $n = 500$ and $k = 480$, the benefit of using subset selection becomes quite small. Section 12.2 gives a greater set of values obtained during our experiments.

Comparison with low-discrepancy point sets: We note that subset selection is not limited to low-discrepancy sequences but can also be used with a low-

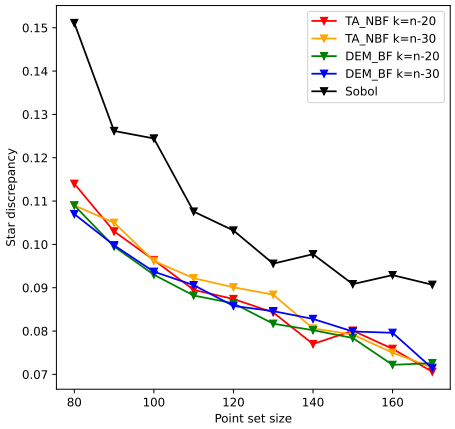

Figure 7.6: Best discrepancy values obtained with our heuristic for $d = 6$ and k=$n - 20$ or $k = n - 30$

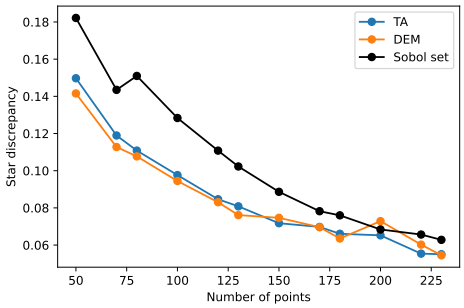

Figure 7.7: Performance of the two subset selection instantiations on Sobol' sets in dimension 6. Subset selection was done for $k = n - 20$, $n - 30$ and $n - 50$ and $n \in \{50, 100, 150, 200, 250\}$.

discrepancy set. One can obtain an n-point set in dimension $d + 1$ from a low-discrepancy sequence in dimension d by taking the first n points and adding i/n as the $d + 1$-th coordinate of the i-th point (see Section 2.2.1). Figure 7.7 shows the results obtained by using subset selection on the obtained sets in dimension 6, starting from the Sobol' sequence. The results are similar as those in the sequence case: $k = n - 20$ and $k = n - 30$ give the best results, while the improvement in the discrepancy value is up to 33%. Finally, the discrepancy values between the set version of Sobol' obtained with the d-dimensional sequence and the $d + 1$-dimensional sequence are quite similar. This suggests that subset selection on *sequences* provides point sets better than low-discrepancy *sets*.

Comparison with random subset selection: Finally, we note that selecting the best subset from a large number of random subsets does not work well. This had been tested extensively when comparing with the exact case in Section 6.4.2, and remains true here. We only provide some simple examples. For $n = 100$ and $k = 80$ in dimensions 4 and 5, 100 000 random subsets give us a best discrepancy of respectively 0.081502 and 0.099460, roughly 10% worse than the DEM with brute force instantiation. This becomes even worse when $n - k$ increases and the number of possible subsets increases: for $n = 100$ and $k = 60$ in dimension 4 the best random subset has discrepancy 0.105830, against 0.087650 with subset selection. While these values do not seem too bad, the main cost of our algorithm is calculating

Table 7.1: Number of points necessary to reach target discrepancies for subset selection and Sobol' in dimensions 4 and 5.

Dimension	Target discrepancy	Sobol' n	Subset selection n
$d = 4$	0.30	15	10
	0.25	17	15
	0.20	28	20
	0.15	45	30
	0.10	89	50
	0.05	201	170
$d = 5$	0.30	17	10
	0.25	26	20
	0.20	38	25
	0.15	52	40
	0.10	112	70
	0.05	255	210

discrepancies. In our experiments, the values were obtained with between 10 000 and 20 000 discrepancy evaluations in the brute force cases and 1 000 and 2 500 evaluations without brute force: far less than the 100 000 required with random subsets to obtain half or a third of the improvement.

7.3.3 Improvements for the Inverse Star Discrepancy

Obtaining point sets with better star discrepancy naturally leads to improvements for the inverse star discrepancy problem mentioned in Section 2.2.2: given a discrepancy target ε, what is the minimal n such that there exists a point set P such that $|P| = n$ and $d^*_\infty(P) \leq \varepsilon$? Table 7.1 shows the improvement in inverse star discrepancy by using the subset selection approach. For applications where the evaluation of each point can take a whole day of calculations, the 12 to 46% gain is substantial. These values were obtained by only taking results from our previous experiments (either in figures or the tables in Section 12.2). We did not try to find the smallest values to reach these discrepancy targets. It is likely that our results could be further improved by a more targeted search. Adapting parameters to the desired instance, increasing the number of runs or removing the 1-hour cutoff are possible options to obtain better results.

We note that the star discrepancy of some known sequences may have been theoretically overestimated, or most likely simply never calculated. In [NW10], Open Problem 42 lists three open questions with targets for the inverse star discrepancy, as well as a conjecture that $n = 10d$ would be a sufficient number of points to reach $d^*_\infty = 0.25$. The open questions had been solved by Hinrichs [Hin13] for the first, and later by Doerr and de Rainville [DR13] for all three,[42] each time by building a new point set. Our experiments on the Sobol' sequence show that $n = 7d$ points are sufficient at least for lower dimensions (smaller than 10). Figure 7.8 shows how the discrepancy of the Sobol' sequence evolves for specific dimensions, as well as a comparison with our subset selection sets.

All three open problems seem to be solved by taking a few hundred points rather than the thousands suggested, confirming de Rainville and Doerr's results without requiring a new set construction. For example, in dimension 15, 146 points have a discrepancy of 0.198, in dimension 30 320 points have a discrepancy of 0.193 and in dimension 25 1205 points have a discrepancy of 0.0996 which can then be lifted to a 50-dimensional point set with discrepancy smaller than 0.2 using Hinrichs' lifting procedure [Hin13]. We acknowledge that these discrepancy values may not be exact as we used the TA heuristic to compute them. However, there is such a large margin both in discrepancy value and number of points that we believe the Sobol' sequence to solve the three problems posed in [NW10].

Figure 7.9 shows the number of points of the Sobol' sequence that are needed to reach discrepancy less than or equal to 0.2 depending on the dimension. This number of points should not be seen as an exact value, but as an upper-bound (with the TA imprecision caveat). Indeed, we found this via binary search to avoid having to compute discrepancy values for all possible n. However, since the star discrepancy of the Sobol' sequence is not monotonous in n (see Figure 7.6), it is possible we have missed better point sets. Nevertheless, we observe a linear relation between the dimension and the number of points, close to $n = 10d$. This reinforces our impression that even the $n = 10d$ conjecture from [NW10] is overestimating the number of points necessary to reach discrepancy ≤ 0.25.

7.3.4 Comparison with the Energy Functional

Section 4.2 introduced a greedy method by Steinerberger to construct a low-discrepancy sequence in one dimension, using the Erdős-Turan inequality. He provides its generalization in higher dimensions in [Ste19] for a point set $X = (x_i)_{i\in\{1,\dots,N\}}$

42 With the caveat that their values could not be verified exactly because of the high dimensions.

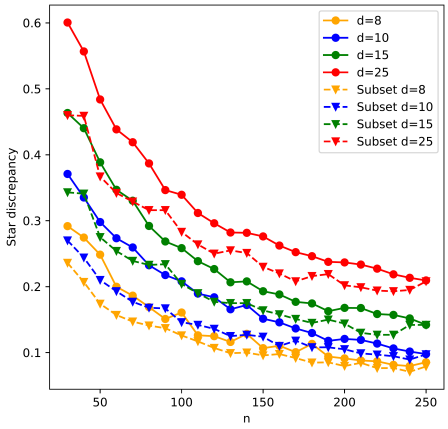

Figure 7.8: Discrepancy values obtained for the Sobol' sequence in different dimensions, compared with the values obtained by subset selection (dashed lines, using the TA heuristic).

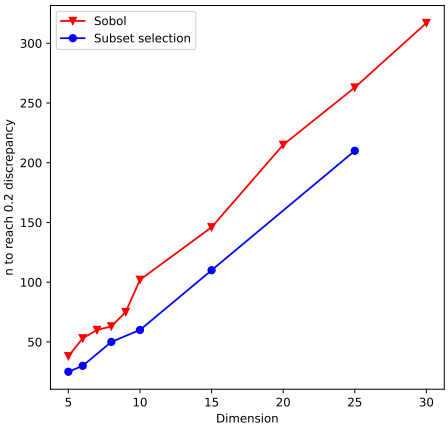

Figure 7.9: Number of points needed to obtain a discrepancy of 0.2 in different dimensions.

.

$$E[X] := \sum_{\substack{1 \le m,n \le N \\ m \ne n}} \prod_{k=1}^{d} (1 - \log(2 \sin(|x_{m,k} - x_{n,k}|\pi))). \tag{7.1}$$

This expression was derived from the Erdős-Koksma-Turán inequality, equation (2.5) introduced in Section 2.2.3, modified to allow the use of gradient descent for optimization. Starting with a given point set of any kind, he applied standard gradient descent until convergence to obtain a new point set which should be better distributed and hopefully have lower discrepancy. He provided a number of examples in dimension 2, while underlining that specific point sets could not be improved by his functional. We ran more extensive experiments in higher dimensions, especially in a setting where n is not necessarily far larger than d. We implemented the functional in C.

Figure 7.10 shows our results and compares the obtained point sets with the results obtained via our subset selection approach. While the energy functional manages to improve in most cases the discrepancy of the input point set, it is much less effective than subset selection. In particular, it is sometimes unable to improve the Sobol' sequence, for example for $d = 5$ and $k = 100$. Subset selection is

Dimension	Set size	Subset Selection		Energy functional		Sobol'
		abs.	Δ to Sobol'	abs.	Δ to Sobol'	
3	50	0.075719	22.00 %	0.092855	4.35 %	0.097075
	100	0.047100	22.25 %	0.060519	0.09 %	0.060575
	150	0.038961	13.10 %	0.041204	8.10 %	0.044834
4	50	0.097189	27.59 %	0.13805	-2.86 %	0.134218
	100	0.061478	33.67 %	0.08828	4.76 %	0.092688
	150	0.053195	13.84 %	0.061526	0.34 %	0.061738
5	50	0.118428	28.44 %	0.157522	4.81 %	0.165488
	100	0.077755	35.58 %	0.128515	-6.47 %	0.120707
	150	0.064438	13.97 %	0.080885	-7.99 %	0.074899
6	50	0.139858	37.99 %	0.210873	6.51 %	0.225548
	100	0.100891	18.93 %	0.127619	-2.55 %	0.124451
	150	0.079571	12.39 %	0.095112	-4.72 %	0.090827
8	50	0.173767	30.09 %	0.24854	0.00 %	0.248547
	100	0.125779	21.78 %	0.160022	0.48 %	0.160793
	150	0.095667	10.35 %	0.133886	-25.46 %	0.106714
10	50	0.209863	29.58 %	0.301397	-1.14 %	0.298001
	100	0.146189	29.73 %	0.216883	-4.24 %	0.208052
	150	0.124148	17.25 %	0.158246	-5.48 %	0.150029

Figure 7.10: Comparison of the energy functional and subset selection. The functional is applied to the Sobol' set of the same size, the subset selection results are taken from our general experiments with $n - k = 50$ for DEM_BF or TA_NBF. We also show percentage improvement of subset selection and the energy functional compared to the Sobol' sequence.

therefore more effective than the functional at creating a new point set with lower discrepancy.

Furthermore, the functional cannot be applied to our new point sets to obtain better point sets. Applying the energy functional optimization to our own low-discrepancy point sets makes them noticeably worse, removing a large part of the initial gain of subset selection. Figure 7.11 gives the discrepancies of point sets obtained by DEM_BF or TA_NBF (n is the nearest multiple of 50 in each line), to which we apply the energy functional to obtain the point sets for the third column. The Sobol' point sets are added as a comparison point in the last column. We observe that the functional makes the point sets noticeably worse, sometimes even worse than the Sobol' point set of corresponding size. This also shows that the energy functional cannot be used as a surrogate for the discrepancy, the point sets obtained with the energy functional approach have much lower energy than those found by subset selection.

Impact of the ground set on the quality of the obtained point sets. Despite this, the energy functional has one clear advantage over our method (apart from the runtime), in that it can take any point set as starting position. While this is not impossible for subset selection, the quality of the starting set strongly limits the quality of the resulting set with our method. For example, Figure 7.12 compares

Dimension	Set size	Subset Selection		Energy functional		Sobol'
		abs.	Δ to Sobol'	abs.	Δ to Sobol'	
4	50	0.097189	27.59 %	0.116502	13.20 %	0.134218
	100	0.061478	33.67 %	0.089109	3.86 %	0.092688
	150	0.053195	13.84 %	0.075875	-22.90 %	0.061738
5	50	0.118428	28.44 %	0.15563	5.96 %	0.165488
	100	0.077755	35.58 %	0.093182	22.80 %	0.120707
	150	0.064438	13.97 %	0.071634	4.36 %	0.074899
6	50	0.139858	37.99 %	0.181417	19.57 %	0.225548
	100	0.100891	18.93 %	0.109322	12.16 %	0.124451
	150	0.079571	12.39 %	0.109322	-20.36 %	0.090827
8	50	0.173767	30.09 %	0.202708	18.44 %	0.248547
	100	0.125779	21.78 %	0.156435	2.71 %	0.160793
	150	0.095667	10.35 %	0.133956	-25.53 %	0.106714
10	50	0.209863	29.58 %	0.2843	4.60 %	0.298001
	100	0.146189	29.73 %	0.216883	-4.24 %	0.208052
	150	0.124148	17.25 %	0.154398	-2.91 %	0.150029

Figure 7.11: Applying the energy functional to our subset selection point sets. These point sets were obtained with $k = n - 50$, see Section 12.2. Also showing percentage improvement of subset selection and subset selection+energy functional compared to the Sobol' sequence of the same size.

the effectiveness of subset selection, the energy functional and a combination of the two on random point sets generated in Python with the random module. For each (n, d) pair, 50 random instances are generated. The Sobol' sets are added for comparison, the energy functional should be compared to the n points line (red) and the two others to the $n - 20$ line (blue). The sets obtained with only subset selection are a lot worse than the low-discrepancy sets, and in the majority of cases worse than those with only the energy functional. However, the combination of the two methods is always at least competitive with the Sobol' set of similar size, and even better in the vast majority of cases. This suggests a new method of computing low-discrepancy point sets, without requiring any knowledge of existing sequences or number theory: starting from any random set, applying successively the energy functional and then subset selection generates good low-discrepancy point sets. It also shows that while the discrepancy of the point sets obtained with the energy functional is not always as good as it could be, the point set created is regular enough to be used as a starting point for subset selection.

7.4 Conclusion and Future Work

Building on the previous chapter on subset selection, we introduced a heuristic that allowed us to obtain better point sets in all dimensions for which the star

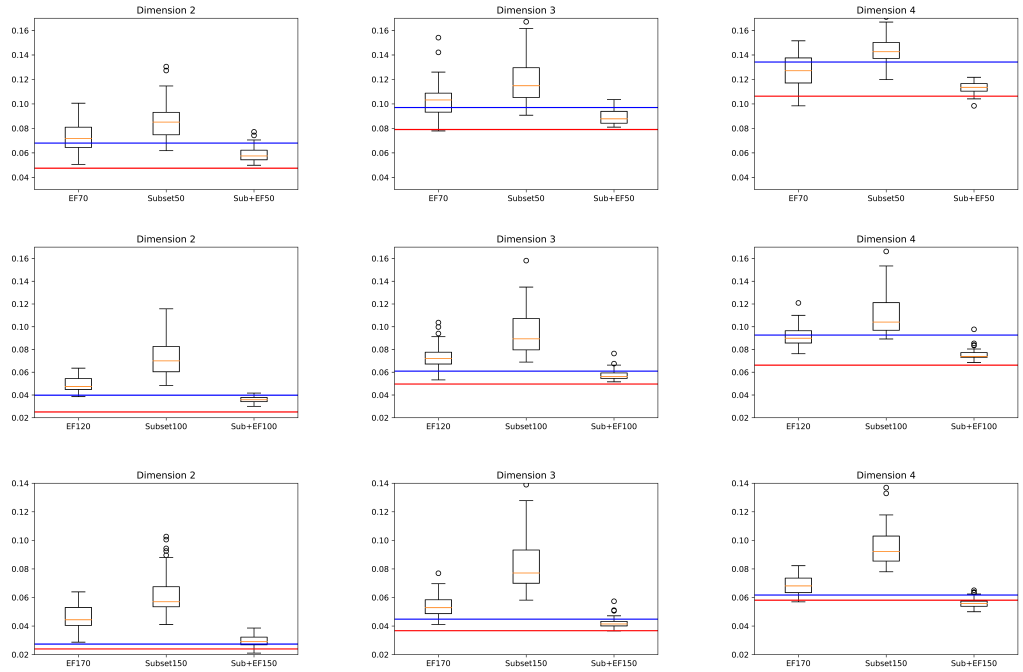

Figure 7.12: A comparison of subset selection (middle in each plot), the energy functional (left in each plot) and the combination of the two methods for random points (right in each plot). This is done in dimensions 2 (left), 3 (middle) and 4 (right) and with an initial $n = 70$ (top), $n = 120$ (middle) and $n = 170$ (bottom). The horizontal lines represent the discrepancy values of the Sobol' sets of relevant size, n (red) and $n - 20$ (blue) in each plot.

discrepancy can be computed, with on average a 20% lower discrepancy than the initial point set. The obtained point sets were compared with known low-discrepancy sequences as well as with an energy functional by Steinerberger. We also provided some initial guidance on the optimal choice of parameters from the problem, with $k = n - 20$ being a good baseline, where k is the subset size and n the input set size.

There are a number of open questions remaining, both on our current heuristic and more general aspects of the problem. Firstly, we have only considered 1-swaps so far. It is likely that heuristics would perform better with more change possibilities at each step. Proposition 7.1 in the next section shows allowing more swaps would not bring better theoretical guarantees and it would entail more computations, but the heuristic would have a better capacity to explore the possible subsets. The second step is also very expensive: determining a limited set of pairs to check

before initiating a restart rather than testing all combinations is likely to improve the heuristics.

Secondly, this problem has shown the limits of current algorithms to compute the star discrepancy. They are expensive and can only give lower bounds when n and d get too high. Some sort of surrogate to replace the star discrepancy evaluations would be extremely useful, as well as interesting in itself to better understand the star discrepancy behavior. As we have shown, Steinerberger's functional would not be good enough for such purposes. A slightly less ambitious goal could be to find a better upper bound for the star discrepancy. Current upper bounds can be obtained either via Thiémard's approach [Thi01a] or bracketing covers [Gne08], neither of which are fast enough for our purposes. Improvements for these algorithms would lead to more precise information on the inverse star discrepancy. Our experiments already seem to show that known sequences perform better than expected, but faster and more precise algorithms would help us refine these conjectures. The promising results of L_2 discrepancy in the next chapter, as well as the ease with which we can evaluate the contribution of each point, suggests that adapting this heuristic to the L_2 discrepancy could bring interesting results.

7.5 Complement: Heuristic Proof

We conclude this chapter with an extra result, showing that any swap-based heuristic for subset selection can be unable to find the global minimum under reasonable hypotheses.

▶ **Proposition 7.1.** Let k-SDSSP-j be the problem of obtaining the minimal star discrepancy subset of size k by only doing improving j-swaps. For every $d \geq 2$, there exist point sets P in dimension d for which the k-SDSSP-j has local minima which are not global minima if $n \geq 2k$ and $j < k$, or $n < 2k$ and $j < n - k$. ◀

Proof. We first consider a base case for $j = k - 1$, $d = 2$, and $n = 2k$. Its extension to all the other combinations of d, j, and n will be described afterwards.

Base case: We construct a point set $P \subseteq [0,1]^2$ of size $2k$ and let $j = k - 1$, represented in Figure 7.13. To build P, we first consider a very small constant α and a set of $k+1$ points $(q_1^{(i)}, q_2^{(i)})$ that satisfy $q_1^{(i)} q_2^{(i)} = 1 - \alpha$. We assume the indices to be sorted such that for all i, $q_1^{(i)} < q_1^{(i+1)}$. We further require that $q_1^{(i)} q_2^{(i+1)} > 1 - 1/(2k)$. These points are the red triangles in Figure 7.13. We then use these $k+1$ points to build the first k points of P, the set $P_A := \{(q_1^{(i)}, q_2^{(i+1)}) : 1 \leq i \leq k\} = \{(p_1^{(i)}, p_2^{(i)}) : i \in \{1, \dots, k\}\}$ given by blue squares in Figure 7.13. From

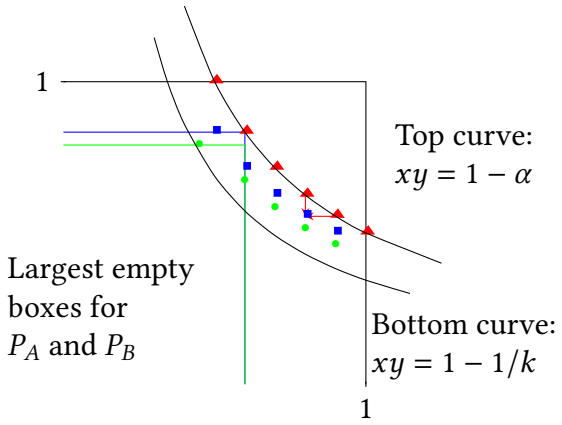

Figure 7.13: An illustration of the different points of the proof of Proposition 7.1: the $q^{(i)}$ are in red, $p^{(i)}$ in blue if $i < k+1$ and green otherwise. The lower curve corresponds to $xy = 1 - 1/k$, the upper one to $xy = 1 - \alpha$. They are not up to scale for readability. The red lines represent how blue points are built, whereas the blue and green boxes are the discrepancy-defining boxes for P_A and P_B respectively. The L_∞ star discrepancy of P_A is $1 - \alpha$ while that of P_B is $1 - \alpha - \delta p_1^{(2)}$. However, it is impossible to transition from P_A to P_B without changing the whole set at once.

each point $p^{(i)}$ in P_A with $i \geq 2$, we can build a point $p^{(k+i)}$, with $p_1^{(k+i)} := p_1^{(i)}$ and $p_2^{(k+i)} := p_2^{(i)} - \delta$, where δ is a small positive constant strictly upper-bounded by $\min_{i \in \{1,\dots,k-1\}} |p_2^{(i)} - p_2^{(i+1)}|$ and such that $p_1^{(k+i)} p_2^{(k+i)} \geq 1 - 1/k$. $p^{(k+1)}$ is such that $p_2^{(k+1)} = p_2^{(1)} - \delta$ and $p_1^{(k+1)} = p_1^{(1)} - \gamma$, where γ is strictly positive smaller than $p_1^{(1)}$ and such that $p_1^{(k+1)} p_2^{(k+1)} \geq 1 - 1/k$. The set formed by these points is defined as P_B, represented by green discs in Figure 7.13.

We first note that for any subset P_k of P of size k, the L_∞ star discrepancy of P_k will be given by the largest box in $[0,1]^d$ containing no points of P_k. Any open box containing points will have local discrepancy at most $1 - 1/k$ (the maximal volume minus the minimal number of points) and any closed box containing points will have local discrepancy at most $1 - (1 - 1/k) = 1/k$ (the maximal number of points minus the minimal volume of a box containing a point). On the other hand, the largest empty box will always have volume at least $1 - 1/k$ (all the points are above the curve $xy = 1 - 1/k$, see Figure 7.13) and thus local discrepancy at least $1 - 1/k$.

We now show that P_A is a local optimum but not a global one. First of all, P_A has discrepancy exactly $1 - \alpha$, obtained for one of the $k+1$ empty boxes whose top-right corner is either $(p_1^{(1)}, 1)$, $(1, p_2^{(k)})$ or $(p_1^{(i+1)}, p_2^{(i)})$ for $i \in \{1, \dots, k-1\}$. By definition of j-swaps, we replace j of the points inside P_A by points in P_B (since

$P_A \cup P_B = P$). Let P_C be our new point set, $|P_A \cap P_C| = 1$. Let $p^{(i)}$ be the point in $P_A \cap P_C$. There are now two different cases to consider:

- If $p^{(k+i)}$ is not in P_C, then the box with upper-right corner in $(p_1^{(i+1)}, p_2^{(i)})$ is still empty. None of the other points could be inside since we have $p_2^{(1)} \geq p_2^{(k+1)} \geq p_2^{(2)} \geq \ldots \geq p_2^{(k)} \geq p_2^{(2k)}$ and the first coordinates are ordered in the reverse order. The discrepancy of P_C is therefore at least that of P_A.
- If $p^{(k+i)}$ is in P_C, then there exists $h \in \{1, \ldots, k\}$ such that $p^{(h)}$ and $p^{(k+h)}$ are not in P_C. The box with upper-right corner in $(p_1^{(h+1)}, p_2^{(h-1)})$ (with $p_1^{(h+1)} = 1$ if $h + 1 > k$ and $p_2^{(h-1)} = 1$ if $h = 1$) has to be empty. By the ordering given above, this empty box will have a volume greater than that of $p_1^{(h+1)} p_2^{(h)}$, the discrepancy of P_A.

By the above, P_A is a local minimum. However, for P_B, the largest empty boxes will have volume $p_1^{(k+i+1)} p_2^{(k+i)}$ for $i \in \{1, \ldots, k-1\}$. By construction $p_1^{(k+i+1)} = p_1^{(i+1)}$ and $p_2^{(k+i)} < p_2^{(i)}$. Since the discrepancy of P_B is given by the largest empty box, we can conclude that $d_\infty^*(P_B) < d_\infty^*(P_A)$, P_A is hence a local minimum for the k-SDSSP-j problem, but not a global one.

Higher dimensions: Taking the same point set with 1's added for all the coordinates in dimensions greater than 2 gives the exact same proof.

More points $n > 2k$: We can add all the $n - 2k$ new points to the region above the curve $xy = 1 - \alpha$. Taking any of these never reduces the volume of the largest empty box, thus a local/global minimum in the base case is still a local/global optimum.

Fewer swaps $j < k - 2$: A local optimum for $j = k - 2$ swaps is still a local optimum for $j < k - 2$ swaps, the global optimum is unchanged.

Fewer points $n < 2k$: The same construction is no longer possible and at least $2k - n$ points have to be shared between any two sets. If $j \geq n - k$ then there are no local optima which are not also global optima, as any subset can be transformed to any other in a single step. If we remove the first $2k - n$ points from P (this is less than n, we are removing only points from P_A), we want the set $P_{LO} := \{p^{(i)} : i \in \{2k - n + 1, 3k - n\}\}$ to be a local minimum. We note that we are keeping the numbering from the base case, i.e, the points in P are numbered from $2k - n + 1$ to $2k$. This requires us to be unable to switch all the $n - k$ "unshared" points to those in P_B (note that $p^{(1)}$ and $p^{(k+1)}$ no longer have a special role as $p^{(1)}$ no longer exists), therefore $j < n - k$. The proof is then the same. ■

8 Greedy Sequence Constructions

The second half of this chapter is based on the preprint [Clé23]. The author is very thankful to Stefan Steinerberger, Carola Doerr and Kathrin Klamroth for some very interesting discussions and questions. The first half is exclusive to this thesis.

8.1 Summary of Results

This section is split into two parts, both describing greedy constructions but with different discrepancy measures. Sections 8.2 and 8.3 describe how to add one point to an existing set such that the L_∞ star discrepancy of the new set is minimal. While it is a relatively natural problem that would be of interest to practitioners,[43] it appears that there does not exist an algorithm for this yet. We provide two approaches, an algorithmic one in Section 8.2 and one based on our non-linear programming approaches from Chapter 5 in Section 8.3. Numerical experiments in Section 8.4 using the second method show that while the methods technically work, the point sets obtained have a higher discrepancy than expected. We believe that greedy L_∞ methods will be more relevant when adding a relatively important number of points to a set, for example to go from n points to $2n$, than when adding a single point.

Sections 8.6 and 8.7 are an empirical study of the Kritzinger sequence introduced in [Kri22]. Limited numerical results in [Kri22; Ste24] suggest it is of similar discrepancy order as traditional low-discrepancy sequences. However, we still lack theoretical results. We first describe a slightly improved algorithm to compute it in one dimension, allowing us to push the number of points computed from the thousands to the millions. We observe that not only does the sequence stay very regular, but also that it appears to outperform the Fibonacci sequence, regardless of the starting point(s). While the properties we rely on for the computation do not hold in higher dimensions, we are still able to provide an exact algorithm to compute it, along with heuristic approaches for higher number of points where the exact methods no longer work. With these, we are able to compute hundreds

43 Given that sets perform theoretically better than sequences, it could be interesting to use a set at first which would then be modified by adding a relatively small number of points if necessary.

(exactly) or thousands of points (approximately) in dimension 2, and hundreds in dimension 3. Despite the imprecision in our heuristics, the resulting sequences still seem competitive with the Sobol' sequence in dimension 2 and the Halton sequence in dimension 3. Based on these results, we conjecture that greedy L_2 minimization leads to low-discrepancy sequences in higher dimensions.

8.2 Greedy Addition of Points: L_∞ Approach

We begin by describing an algorithm to greedily add a point to an existing set with respect to the L_∞ star discrepancy in dimension 2. We will highlight during the algorithm description why it cannot be easily adapted to higher dimensions. The algorithm in dimension 1 is much simpler and will not be described in detail here.[44]

The problem we are trying to solve is the following. Given a point set $P \in ([0,1]^2)^n$, we want to determine

$$x_{n+1} := \operatorname*{arg\,min}_{y \in [0,1)^2} d^*_\infty(P \cup \{y\}).$$

This problem naturally arises when one is working with a set of low-discrepancy points, and wants to add further points without redoing all the previous experiments. While working with a sequence is the more natural approach, one may not know this in advance and would like to benefit from the better regularity of sets. It is also a simple algorithmic approach to try constructing a low-discrepancy set, as was done to initialize our branch-and-bound approach in Chapter 6.

This section describes an algorithmic method of solving this problem, while Section 8.3 gives an optimization approach based on the methods introduced in Chapter 5.

For the algorithmic approach, when adding a new point (q_1, q_2), we need to consider all the boxes with top-right corner in $\overline{\Gamma}(P \cup \{q\})$ which could define the new discrepancy value. These can be of the following types:

- $[0, p)$ or $[0, p]$ where $q \notin [0, p)$. This is done in Section 8.2.1 via $v(b)$.

44 Supposing the points are ordered with dummy points $0 = x^{(0)}$ and $1 = x^{(n+1)}$, we only need to look at each interval $[x^{(i)}, x^{(i+1)})$. Wherever we place the point in the interval, it will contain i points if open and $i + 1$ if closed, which can be solved directly. A similar dynamic programming method as the one we will describe in this section, but only in one dimension, can easily compute the remaining discrepancy values for the untouched intervals. There only remains to compare which interval gives the best value.

- $[0, p)$ or $[0, p]$ where $b \in [0, p)$. This is also done in Section 8.2.1, this time via $w(b)$.
- $[0, p)$ or $[0, p]$ with $p_1 = q_1$ or $p_2 = q_2$. This is done in Section 8.2.2.
- $[0, q]$. This box leads to a parabolic surface and is considered at the end of Section 8.2.2.

In both the following sections, we will be considering the decomposition of $[0, 1]$ based on the grid $\overline{\Gamma}(P)$. We will be working with boxes $[a, b)$ of this grid, where $a := (a_1, a_2)$ and $b := (b_1, b_2)$ are such that there is no other coordinate between a_1 and b_1, or between a_2 and b_2. In Section 8.2.1, we study how local discrepancies change for boxes in $\overline{\Gamma(P)}$, i.e., which are not defined by the new point added. Section 8.2.2 will then describe how to optimally add a point to a box $[a, b)$. The solutions of both steps are then regrouped to find the optimal point to add.

8.2.1 Boxes Without the New Point

We suppose the new point will be added to a box $[a, b)$ defined by the grid of $\overline{\Gamma}(P)$. Choosing to place a point inside this box guarantees that we will not place it in any of the smaller boxes, and it will be inside all of the bigger boxes. This means the discrepancy of all the other boxes will change and this needs to be taken into account in order to be able to make the best choice between the different boxes of the grid of $\overline{\Gamma}(P)$. For each box corner $b \in \overline{\Gamma}(P)$, one can keep two values $v(b)$ and $w(b)$. $v(b)$ represents the worst discrepancy value in boxes of $\overline{\Gamma}(P)$ that *do not dominate* b if the point is placed in the box $[b, 1]$. Similarly, $w(b)$ represents the worst discrepancy value in boxes *dominating* b if the point is placed in $[0, b]$. All other boxes in the new grid $\overline{\Gamma}(P \cup \{q\})$ are either examined in the next step or do not need to be examined as they are not critical.

Let us write $\overline{\Gamma}(P) := X \times Y$, where $X := \{x_1, \ldots, x_n, 1 : x_1 \leq x_2 \ldots \leq x_n < 1\}$ and $Y := \{y_1, \ldots, y_n, 1 : y_1 \leq y_2 \ldots \leq y_n < 1\}$. Barring edge cases where one of the terms will be missing (one can set them to 0 if undefined), we have

$$w(x_i, y_j) = \max\Big(w(x_{i+1}, y_j), w(x_i, y_{j+1}), \overline{\delta}\big((x_i, y_j), P \cup \{(0,0)\}\big), \\ \delta\big((x_i, y_j), P \cup \{(0,0)\}\big)\Big). \tag{8.1}$$

In other words, the worst discrepancy value in boxes dominating (x_i, y_j) is either reached in the box itself or by a box dominating either (x_{i+1}, y_j) or (x_i, y_{j+1}). $P \cup$

$\{(0,0)\}$ represents the fact that the box $\big[(0,0),(x_i,y_j)\big)$ will always[45] contain the new point, wherever we actually choose it to be. This can be solved with a simple dynamic programming algorithm, with the base case being for $x_i = y_j = 1$.

One will notice that we can introduce $\overline{w}(b)$ as the worst discrepancy value in boxes *dominated* by b if the point is placed in $[b,1]$. With a symmetric dynamic programming method,[46] we can solve $\overline{w}(b)$ over the whole grid. It suffices to notice that $v(a) = \max(\overline{w}(a_1,1),\overline{w}(1,a_2))$ to compute $v(a)$ over the grid. Both the dynamic programming algorithms are in $O(n^2)$, although they cannot be done at the same time.

8.2.2 Optimal Placement Inside a Box of $\overline{\Gamma}(P)$

We now consider the problem of optimally placing a point inside a single half-open box of the grid associated with $\overline{\Gamma}(P)$, $[a,b)$ where $a := (a_1,a_2)$ and $b := (b_1,b_2)$.

For any such box $[a,b)$, the number of points contained inside $[0,q)$ is the same regardless of the choice of $a \leq q < b$. Optimally placing a point inside this box requires to consider *only* the critical boxes using a coordinate of q (i.e., a box of the shape $[0,p)$ or $[0,p]$, where $p_1 = q_1$ or $p_2 = q_2$) and the closed box $[0,q]$. Indeed, for all the other boxes, regardless of the choice of q, the boxes will contain a fixed number of points and therefore their local discrepancies were computed in the previous step (see Section 8.2.1).

We first consider the boxes $[0,p)$ or $[0,p]$ where $p_1 = q_1$. A toy example is provided in Figure 8.1.

For the closed boxes, each of them defines a local discrepancy equation of the shape $y = N_i - q_1 c_{2,i}$ where N_i is the number of points inside $[(0,0),(q_1,c_{2,i})]$, $c_{2,i}$ is the second coordinate of another point in the set and y the discrepancy value, function of q_1. We only consider critical boxes above our current box (see Figure 3.2 and the definition of critical boxes, and Figure 8.1 for an example). Let us call $c_{2,1},\dots,c_{2,k}$ the coefficients for the different equations, corresponding to the second coordinates of the other points. On the interval $[a_1,b_1)$, we want to find the highest discrepancy value given by *any* of these equations.[47] In other words, we are looking

45 This would be incorrect in the specific case that the optimal point inside the box shares a coordinate with b, i.e. when we solve the previous problem in $[a,b)$ and not $[a,b-\varepsilon)$. If we ignore the issue with the imprecise line equations mentioned in the next subsection, the last term in Equation (8.1) should be $\delta\big((x_i,y_j),P\big)$. This does not present any extra technical difficulties.

46 The base case is now in $(0,0)$, rather than $i+1$ it is $i-1$, etc...

47 There is a technical difficulty with the open interval due to computer finite precision. In our case, it is more convenient to accept a discretization step, i.e. we are working in $[a_1,b_1-\varepsilon]$, than obtaining a supremum. This supremum could indeed lead to an incorrect point count in

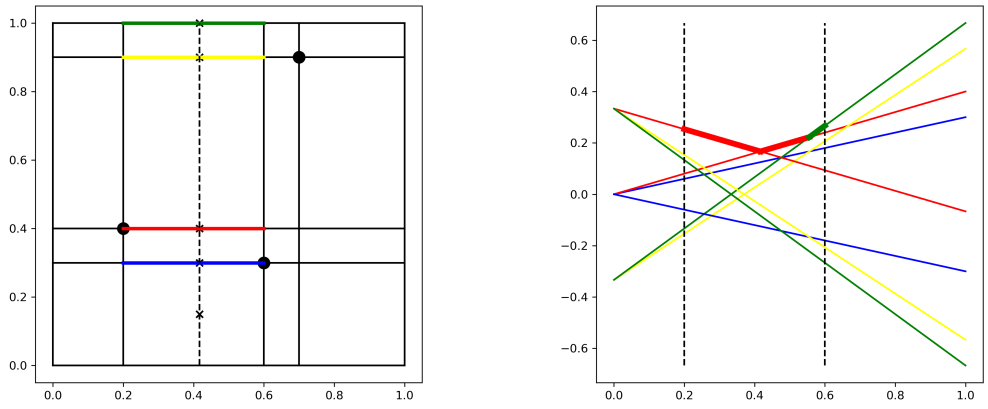

Figure 8.1: On the left, we have the grid $\overline{\Gamma}(P)$ associated to a point set with three points (black circles). We want to place a point in the second box in the lowest row. This new point will define new critical boxes at the intersection with all the grid lines above it. Each of the coloured line segments leads to two line equations, represented on the right. The decreasing lines correspond to the open boxes, and the increasing ones to the closed boxes. We want to find the upper convex hull of these lines on the interval $[0.2, 0.6]$. This is represented by the thicker lines. The lowest discrepancy value that can be obtained depending on x is the lowest point on these thicker lines, here $x = 5/12$. One would need to do a similar process with the grid lines to the right, then compare with the parabolic surface associated with the closed box $[0, q]$, where q is the new point, to finally obtain the optimal choice in the box.

for the upper convex hull of a line arrangement of decreasing lines. We also have an extra advantage in that we know in advance the $c_{2,i}$ values and their ordering. This is a classical problem, solved in [Ata86]. We give a description of the method as we are in a particularly simple case where the complexity is linear in the number of lines.

We start with an empty list of intersections M.[48] The line with steepest slope is the highest one at $-\infty$, we can then find its intersection point with the line with the second steepest slope. If this intersection point is above b_1, it can be forgotten, otherwise we update our list of intersections. All lines from the steepest to the

the equations as we could count a point with a coordinate equal to b_1 or b_2, and is in any case considered in another box.

48 We will store the successive equations forming the convex hull as well as the intersection points. For ease of reading, we will refer only to this as the list of intersections. There is no extra cost in obtaining the information required to maintain both at the same time.

flattest are considered in this manner. At each step, we find the intersection with the last line to be part of the intersection list M.[49] If this intersection is before the *previous* intersection in our list, we forget the last line in M and re-do this step. If it is after and below b_1, we add it in the list. Since each intersection comparison leads to either an addition of a line to M or the definitive removal of one, there are at most $2k$ such steps, leading to an $O(k)$ complexity, itself in $O(n)$.

One can notice that for the open boxes $[0, p)$, we will have the same problem, on the same interval, but with a set of increasing lines. One can therefore solve this with a similar complexity. We now need to find the upper convex hull of the union of these two sets of line arrangements. Since one consists of increasing lines and the other of decreasing lines on the same interval and both are sorted, we simply need to go through the intersection points from a_1 to b_1 until one of the increasing lines gives a higher discrepancy value than the decreasing lines. We then use the equations for the two locally dominant lines to find the exact intersection point. By keeping the beginning of the list for the decreasing lines up to that intersection point and the end of the list of increasing lines after that intersection point, we have access to the exact discrepancy value over the interval $[a_1, b_1)$ depending on q_1, as well as which box gives this value (and the associated line equation), and where the boxes change.

This method gives us the list M_1 of intersection points and associated segments that define the discrepancy value over $[a_1, b_1)$. Similarly, we can obtain the list M_2 for $[a_2, b_2)$. For each box in the grid defined by $M_1 \times M_2$, there remains to merge the results and find which equation locally defines the discrepancy. One should also not forget to include the discrepancy for the box $[0, q]$. Overall, we need to compare the planes containing the segments found previously and that are constant in y for M_1 (respectively x for M_2), as well as the parabolic surface defined by $z = xy - N_a$ where N_a is the number of points in $[0, a]$. Since we have all the necessary equations, this can be done in constant time for each box of the grid $M_1 \times M_2$.[50] Overall, this step is in $O(n^2)$ time as there are $O(n^2)$ elements in $M_1 \times M_2$ to examine. This enables us to find $u(a, b)$, the best discrepancy value that can be obtained when placing a point in $[a, b)$ for boxes defined by this point, as well as the associated point q.

There only remains to regroup the two steps. When adding a point to a box $[a, b)$, the local discrepancy of the critical boxes will either appear in $v(a)$, $w(b)$

49 One can do this in constant time with a stack.

50 For the parabolic surface one could first find the minimal point with the two planes and verify if it is indeed above the parabolic surface to avoid having to compute the equation of the intersection.

or $u(a,b)$. In other words, the optimal point to add is exactly the point associated with the following minimum

$$\min_{(a,b)\in\overline{\Gamma}(P)} \max(v(a), w(b), u(a,b)).$$

In the worst case, this takes $O(n^4)$ time as there are $O(n^2)$ such boxes $[a,b)$. However, in practice, one can bound $u(a,b)$ for all the different $[a,b)$ by lower-bounding the discrepancies that can be obtained when adding a point in any box $[0,p)$ or $[0,p]$. To give an example, for $q \in [a,b)$, any open box $\left[0,(q_1,c_{2,i})\right)$ will contain as many points as $[0,(a_1,c_{2,i})]$ and a larger volume. By passing over all boxes of $\overline{\Gamma}(P)$, both open and closed, one can obtain bounds for all boxes $[a,b)$ in $O(n^2)$ time. Both these bounds and the $v(a)$ and $w(b)$ values can therefore allow us to avoid having to compute $u(a,b)$ in every box.

Finally, there are a number of tricks to further improve the complexity in practice. Firstly, a lot of the lines in the line arrangements will be in common for the same row/column of the grid. With an extra memory cost, one can compute all the line arrangements for the open boxes for a given row/column at once, and similarly for closed boxes. There will still be the merge step in any case. We also expect that for a large majority of boxes the parabolic surface is always either below or above the line arrangement's convex hull, and no computations are required in the second case. This should be the case in particular for slightly unbalanced sets, for example those where an overfilled box of the old set will still be the worst box of the new set even if the new point is placed outside. Finally, if we have multiple boxes with the same point count, only one of these can appear in the convex hull. This further decreases the potential size of $M_1 \times M_2$. Overall, using the previous bounds and these technical comments, we expect the complexity to be a lot smaller than $O(n^4)$ in practice ($\Omega(n^2)$ is certain).

8.3 An Optimization Perspective

Greedily computing the optimal point for the L_2 star discrepancy can also be seen as a variant of our optimization models introduced in Section 5.2. Indeed, we can consider that we are trying to optimize the placement of a set of $n+1$ points, while fixing the variables associated to n of these points.[51] This can be done in both models, but we will only present the continuous version. The only aspect one should be careful with is the ordering: we can no longer suppose all the points are

51 Clearly, this can be generalized to adding j points to an existing set for any $j > 0$.

sorted in the first dimension since we do not know where our $n+1$-th point will be relative to the n fixed points. The reasoning behind the constraints is generally the same as in model (5.5), and we refer the reader to Section 5.2.2.

We define $r_{i,j}$ the binary variable equal to 1 if and only if the new point (x_{2n+3}, x_{2n+4}) is dominated in the first variable by $x^{(i)}$ (this dominance relation is itself defined by variable v_i) and in the second by $x^{(j)}$ (similarly defined by variable w_j). Given a fixed set $P := \{x^{(i)} : i \in \{1, \dots, n\}\}$, we obtain the following model.

$$\min \quad f$$

$$\text{s.t.} \quad \frac{1}{n}\left(r_{i,j} + \sum_{u=1}^{i} y_{uj}\right) - x_{2i-1}x_{2j} \le f + (1 - y_{ij}) \qquad \forall i, j = 1, \dots, n,\ j \le i \tag{8.2a}$$

$$\frac{-1}{n}\left(r_{i,j} + \sum_{u=0}^{i-1} y_{uj} - 1\right) + x_{2i-1}x_{2j} \le f + (1 - y_{ij}) \qquad \forall i = 1, \dots, n+1,\ j < i \tag{8.2b}$$

$$\frac{1}{n}\left(1 + \sum_{u=1}^{i} (1 - w_u)\right) - x_{2i-1}x_{2n+4} \le f + w_i + (1 - v_i) \qquad \forall i = 1, \dots, n+1 \tag{8.2c}$$

$$\frac{-1}{n}\left(\sum_{u=1}^{i-1} (1 - w_u)\right) - x_{2i-1}x_{2n+4} \le f + w_i + (1 - v_i) \qquad \forall i = 1, \dots, n+1 \tag{8.2d}$$

$$\frac{1}{n}\left(1 + \sum_{u=1}^{n} y_{uj}(1 - v_u)\right) + x_{2n+3}x_{2j} \le f + v_j + (1 - w_j) \qquad \forall j = 1, \dots, n+1 \tag{8.2e}$$

$$\frac{-1}{n}\left(\sum_{u=1}^{n} y_{uj}(1 - v_u) - 1\right) + x_{2n+3}x_{2j} \le f + v_j \qquad \forall j = 1, \dots, n+1 \tag{8.2f}$$

$$\frac{-1}{n}\left(\sum_{u=1}^{n} (1 - r_{u,u})\right) + x_{2n+3}x_{2n+4} \le f \tag{8.2g}$$

$$x_{2j} - x_{2i} \ge y_{ij} - 1 + \varepsilon \qquad \forall i, j = 1, \dots, n,\ i < j \tag{8.2h}$$

$$x_{2j} - x_{2i} \le y_{ij} \qquad \forall i, j = 1, \dots, n,\ i < j \tag{8.2i}$$

$$y_{ij} = 1 - y_{ji} \qquad \forall i, j = 1, \dots, n,\ i > j \tag{8.2j}$$

$$y_{ii} = 1 \qquad \forall i = 1, \dots, n \tag{8.2k}$$

$$x_{2n+4} - x_{2i} \geq w_i - 1 + \varepsilon \qquad \forall i, = 1, \dots, n,\ i \tag{8.2l}$$

$$x_{2n+4} - x_{2i} \leq w_i + \varepsilon \qquad \forall i = 1, \dots, n,\ i \tag{8.2m}$$

$$x_{2n+3} - x_{2i-1} \geq v_i - 1 + \varepsilon \qquad \forall i, = 1, \dots, n,\ i \tag{8.2n}$$

$$x_{2n+3} - x_{2i-1} \leq v_i + \varepsilon \qquad \forall i = 1, \dots, n,\ i \tag{8.2o}$$

$$r_{i,j} \geq v_i + w_j - 1 \qquad \forall i, j = 1, \dots, n \tag{8.2p}$$

$$r_{i,j} \leq v_i \qquad \forall i, j = 1, \dots, n \tag{8.2q}$$

$$r_{i,j} \leq w_j \qquad \forall i, j = 1, \dots, n \tag{8.2r}$$

$$x_0 = x_{2n+1} = 1$$

$$y_{0j} = 0 \qquad \forall j = 1, \dots, n$$

$$y_{j0} = y_{(n+1),j} = 1 \qquad \forall j = 0, \dots, n$$

$$x_i = x_1^{((1+i)/2)} \qquad \forall i = 1, 3, \dots, 2n-1$$

$$x_i = x_2^{(i/2)} \qquad \forall i = 2, 4, \dots, 2n$$

$$y_{ij}, r_{i,j}, v_i, w_j \in \{0, 1\} \qquad \forall i, j = 1, \dots, n$$

$$f \geq 0.$$

Apart from the equations defining the v, w and r variables, we also need to modify all the local discrepancy constraints. Indeed, for the boxes defined by the old points, equations (8.2a) and (8.2b), we also need to check if the new point is in the box or not. This is directly known with variable $r_{i,j}$. The boxes defined on one side by the new point lead to equations (8.2c), (8.2d), (8.2e) and (8.2f). For these, we know if the box is critical by looking at w_j and v_j, or v_i and w_i. The point count is adapted using both the sorting of the initial points and our new variables. We also instantly know if the new point will be inside by the criticality check on the right-hand side: if the box is critical, the point is inside the closed box, and not in the other cases. Finally, equation (8.2g) counts the number of points inside the closed box defined by the coordinates of the new point.

8.4 Experimental Results

While we are able to solve this problem for much higher n than in Chapter 5, the usefulness of this method remains uncertain. Indeed, repetitively adding a point to the same set, for example to go from 8 to 16 points, gives a relatively poor set. Even

worse, starting from a single point in $(0.5, 0.5)$, as we will do for the Kritzinger sequence in the next step, can potentially lead to a very bad set. Table 8.1 describes the next point added by the solver and the new discrepancy value for the resulting set. Figure 8.2 plots the discrepancy of the new set if one adds a point in any position at each of the four steps given in Table 8.1. It is obvious that one does not want points to be so close to each other, yet the chosen points are in the correct zone. This highlights the need for extra criteria when choosing where the next point should go. We stress that greedy L_∞ constructions are not systematically bad, but that more work is necessary to use them well than for greedy L_2 constructions (see Section 8.6 and Section 8.7.2).

Table 8.1: Successive discrepancy values obtained when adding a point starting with the set $(0.5, 0.5)$. All the points added by the solver are at the edge of the optimal zone but remain correct.

New point	(0.9999,0.5)	(0.0,0.5)	(0.5,0.5)	(0.6,0.5)
Discrepancy	0.5	0.5	0.5	0.5

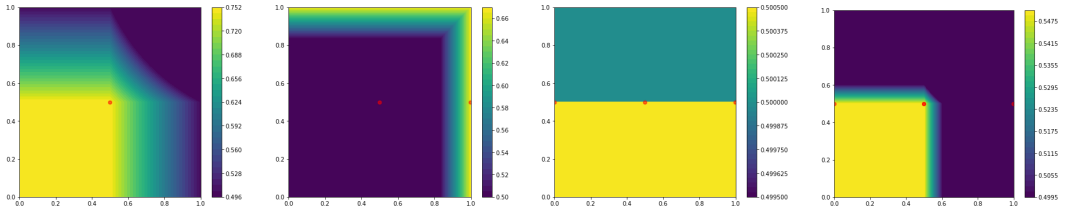

Figure 8.2: Each plot describes the new discrepancy value of the set that would be obtained if we added a point in any position. The four images from left to right correspond to four successive steps of the greedy approach starting with a single point. The points appear in red, always on the $y = 0.5$ line. Two points are in $(0.5, 0.5)$ in the last image. The next point is always picked in the darkest zone, while for the third image any point picked would lead to the same discrepancy value.[52] It would be possible to add an infinite number of points on the line $y = 0.5$. Indeed, adding a point above means there remains an empty box of volume 0.5 and below means there is a closed box of volume 0.5 containing all the points.

While this is a worst-case example, starting with a good set does not lead to much better results. Table 8.2 shows the discrepancy values obtained when starting from our optimal point set of size 8. While some choices can be good (twelve to

52 The imprecision comes from our minimal distance between two points in the mathematical programming formulation, and from the discretized calculation when plotting in Python.

thirteen points), the overall value seems relatively stable. More importantly, it is much higher than the initial set which had a discrepancy of 0.1328.

Table 8.2: Successive discrepancy values obtained by greedily adding one point from the 8-point optimal set. This set had a discrepancy of 0.1328.

n	9	10	11	12	13	14	15	16
$d^*_\infty(P)$	0.1745	0.2078	0.1927	0.2078	0.1577	0.1602	0.1745	0.1893

In a similar way, successively adding then removing one point from a given set in the hope of improving the discrepancy of the set gives generally poor results (the same toy-example starting with $(0.5, 0.5)$ applies again).[53] One of the reasons is that there is no single optimal point to add in the majority of cases, if we only consider the L_∞ star discrepancy of the resulting set. Indeed, as long as the worst box keeps the same discrepancy value, we can make many other boxes worse while not changing the overall value. This lack of global knowledge makes the method relatively poor. Adding penalties in the objective function, for example when placing a point too close to another, could prove to be an acceptable correction.

Another possible method to improve the results of the greedy L_∞ approach could be to add *multiple* points at the same time and not just one. This can be done with a very similar model as model (8.2): rather than adding a single point associated with variables x_{2n+1} and x_{2n+2}, we will add m points associated with variables x_{2n+1} to x_{2n+2m}. All the binary variables can naturally be extended, as well as the definitions of the constraints. With this extended model, we are able to add up to around 8 points at a time to an existing set. Unsurprisingly, the results are much better than when adding step by step, as shown in Figure 8.3. From a runtime perspective, while it is still much faster than solving the entire model for 16 points, we are limited in how many points at once we can add. Based on some experiments on Gurobi on a laptop, eight seems to be the limit. Should one want to run this multiple times in a row to build a bigger set, four or six extra points might a better choice than higher m.[54]

Overall, while greedy L_∞ constructions perform badly, there is still hope in their

53 A very natural family of poor examples is those of point sets with one clearly worse box, for example with too many points. Any point outside this region will be optimal, including those that could be very close to existing points. This is what happens with the sets taken from the Sobol' sequence and its overfilled boxes as shown in Figure 1.1. The new point is placed in the top-right corner, often on an existing point, greatly limiting the improvement potential.

54 Preliminary experiments where we add extra constraints limiting the search space have given good results and run much faster, for example by imposing that our new points alternate coordinate-wise with our old points. They do not have the same optimality guarantee however.

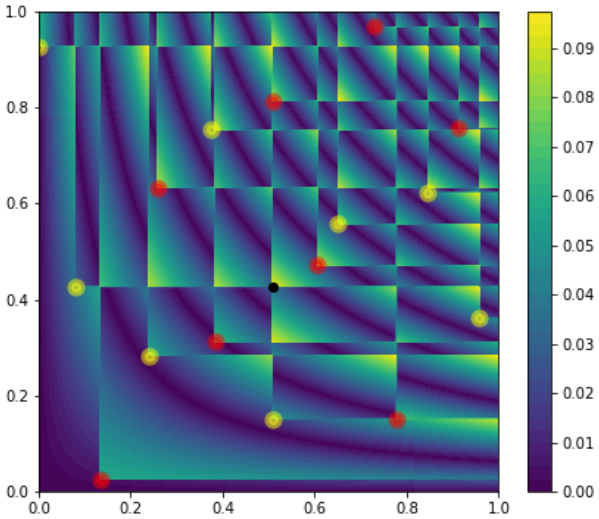

Figure 8.3: One of the optimal 16 points sets, with the constraint that it must contain the 8 red points. The 8 yellow points are those added. The discrepancy is much closer to the optimal one, 0.0739, than when adding one by one as in Table 8.2.

use with adapted algorithms. Adding multiple points at once is a possibility to limit the numerous possibilities available when picking a single point and, more importantly, adding secondary objectives or constraints to enforce additional geometric properties seems promising. Choosing these objectives well, in particular with the aim of repeating multiple times the greedy step, remains an open question.[55]

8.5 The Kritzinger Sequence: Summary of Results

We now turn our attention to the more promising greedy approach, the construction of the Kritzinger sequence as introduced in Section 4.3. In other words, rather than greedily adding the optimal point with respect to the L_∞ star discrepancy, we add the optimal point with respect to the L_2 star discrepancy. The numerical experiments in [Kri22; Ste24] are limited to a few thousand points in dimension 1,

55 Initial experiments with a secondary objective maximizing the minimal distance between points or between coordinates of different points are promising. This could simply be because they avoid the solver issue of bringing points close together when there is freedom of choice. In any case, these simple objectives avoid the biggest pitfalls highlighted previously.

a little bit above 1500 based on plots in [Kri22; Ste24], and no construction methods are provided for higher dimensions. We introduce in the next sections a better algorithm to compute sequences of up to a few millions of points in dimension 1 (Section 8.6). We also provide a set of numerical methods to construct, exactly or approximately, the Kritzinger sequence in dimensions 2 and 3 (Section 8.7).

In dimension 1, we obtain much stronger evidence that the L_∞ star discrepancy of the sequence is of order $O(\log(n)/n)$. In dimensions 2 and 3, we show that we can build a large number of approximate sequences that can rival low-discrepancy sequences, strengthening the conjecture that the discrepancy of the Kritzinger sequence in dimension d is $O(\log^d(n)/n)$. Extending observations made in [Ste19], we also show in Section 8.6.2 that the Kritzinger sequence can be initialized with a very bad set of points and still rival the Fibonacci sequence.[56] Our code as well as the points generated are available at [Clé].

8.6 Competitiveness of the Kritzinger Sequence in Dimension 1

8.6.1 Generating More Points

As mentioned in the general setting in Section 4.3, in one dimension, after scaling by $n+1$, the function we want to minimize is

$$F(y,P) = -(n+1)(1-y^2) + (1-y) + 2\sum_{x\in P}(1-\max(x,y)).$$

If we consider $x_1,\dots,x_n$ to be the ordered points of P, $F(y,P)$ is a second order polynomial on each $[x_i,x_{i+1}]$ interval (where $x_0 = 0$ and $x_{n+1} = 1$ are added as dummy points). More specifically, for the interval $[x_i,x_{i+1}]$, we have $F_i(y,P) := (n+1)y^2 - A_i y + B_i$, where $A_i = 2i+1$ and B_i is a constant. The minima of such polynomials will be reached only for $y = (2i+1)/(2(n+1))$, where $i \in \{0,\dots,n\}$ [Kri22, Theorem 2]. It is therefore only necessary to check these $(n+1)$ candidates to find the global minimum. While the minimum may not be unique, there will be no large zone of points all reaching this minimal value, but only a few localized points (a single one in the majority of cases).

56 This was also emphasized by Stefan Steinerberger during his presentation in the Dagstuhl seminar 23351, as well as during subsequent discussions. We thank him for the help and encouragement provided for this work.

Importantly, we can derive the expression of F_{i-1} from that of F_i. Indeed, F_{i-1} differs from F_i only for the term involving x_i which became $1 - x_i$ rather than $1 - y$. We obtain $A_{i-1} = A_i - 2$ and $B_{i-1} = B_i - 2x_i$. Since the last polynomial is known, given by $F_n(y, P) = (n+1)y^2 - (2n+1)y + n$, each polynomial and its solution can be found in constant time. Since there are $O(k)$ such polynomials for the addition of the k-th point, this gives an $O(n^2)$ runtime for the greedy construction of the first n points of a Kritzinger sequence. We remark that keeping the points sorted can be done in constant time, this has no impact on the overall complexity.

We note that this is theoretically better than Remark 2 from [Kri22], which says that if an interval $[\ell/n, (\ell+1)/n]$ already contains a point then it will not contain the next point. Indeed, checking if each interval already contains a point takes constant time, with an extra linear time to compute the local solution. If there are multiple empty intervals, this will take more time. We can combine the two together but the gains are negligible.

Using this method, we provide the first million points of the Kritzinger sequence in Figure 8.4. We compare them with one of the best traditional low-discrepancy sequences, the Fibonacci sequence.[57] We also include the upper bound on the best asymptotic constant by Ostromoukhov [Ost09] in the plots. The sequence that gives this constant is compared to the Kritzinger sequence on the right in Figure 8.4. While the Ostromoukhov sequence performs much better for very specific values, the Kritzinger sequence outperforms the Ostromoukhov sequence in the vast majority of cases.

Figure 8.4 shows that the Kritzinger sequence is competitive with, if not on average better than, the golden ratio Kronecker sequence for the first 1 000 000 points, with a discrepancy value computed every 1 000 points. Plots with one-point steps in Figure 8.5 show that the spikes in the Kritzinger sequence's discrepancy values are extremely localized. We note that the big spikes represent only *one or two* successive subpar point choices out of thousands of points. Figure 8.6 describes the proportion of instances for which the golden ratio Kronecker sequence is better than the Kritzinger sequence. These experiments show the Kritzinger sequence is reliably outperforming the Fibonacci sequence, being better for around 65% of the instances. This empirical convergence provides further evidence for the following conjecture by Kritzinger.

▶ **Conjecture 8.1.** [Kri22, Conjecture 2] The Kritzinger sequence has an L_∞ star discrepancy in $O(\log(n)/n)$ in one dimension. ◀

57 As described in Section 2.3, it is given by $x_n = \{n\alpha\}$ where $\{\cdot\}$ represents the fractional part and α is the golden ratio $(1+\sqrt{5})/2$.

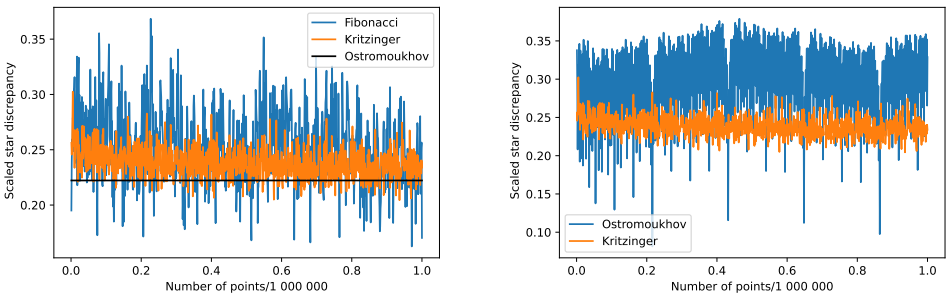

Figure 8.4: Comparison of the first million points of the Kritzinger sequence with the Fibonacci sequence (left) and the Ostromoukhov sequence (right). Values are calculated every 1 000 points and scaled by $n/\log(n)$. The black line corresponds to the best theoretical upper bound on the asymptotic discrepancy constant by Ostromoukhov. We clearly notice that this asymptotic constant is much better than the discrepancy of the Ostromoukhov sequence for a million points. While the asymptotic discrepancy order of the Ostromoukhov sequence is the best known to this day, it is outperformed by the Kritzinger sequence for the first million points.

8.6.2 Changing the Initialization Point(s)

Impact of the initial point: In all our work so far, the sequence was naïvely initialized with a single point in 0.5. We performed experiments with $x_0 = i \times 0.1$, for $i = 0$ to 9 and an extra $x_{10} = 0.9999$. All the sequences performed similarly as Figure 8.7 shows. For readability, we plot the minimum, maximum, and average scaled discrepancy for each of these sequences. All of these have similar behaviors and perform very well.

Changing this single initialization point to 5 randomly selected values has also no noticeable impact on the quality of the obtained sequence, as shown in Figure 8.8. We note that both in this case and the previous single point initialization, there is no meaningful difference between the different runs. All seem to have the same behaviour, there is not one that performs better more often or another that is more often worse.

Correcting a bad initial set: Finally, we initialize the sequence with a large set of badly chosen points: 100 points from 0 to 0.01 with a step-size of 10^{-4}. While the discrepancy values are unsurprisingly relatively poor initially, they are as good as those obtained with a single initial point at the 10 000 point mark, and continue to perform just as well as the previous sequences afterwards, as Figure 8.9 shows. The initial values are not included in the plot to make it readable. The first 9 values (for $n = 1000$ to $n = 9000$) nevertheless come very close to the discrepancy that would

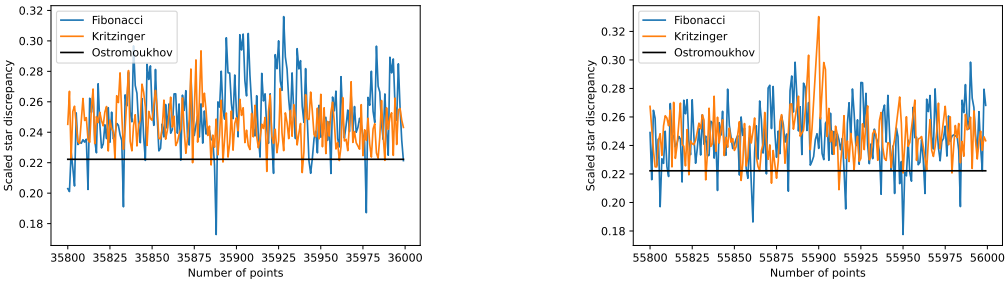

Figure 8.5: Localized plots in regions where the Kritzinger sequence was seemingly not performing well. Discrepancy values are calculated for all n and scaled by $n/\log(n)$. Higher spikes in the Kritzinger sequence correspond to very few badly chosen points and are quickly corrected by the next choices.

be obtained with only the initial points in the box $[0, 0.01)$. For example, for 7000 points, we obtain a discrepancy of 0.00438 whereas any set of 7000 points with 100 points in $[0, 0.01]$ has a discrepancy of at least 0.00428.

8.7 Evaluating the Kritzinger Sequence in Higher Dimensions

8.7.1 Construction of the Kritzinger Sequence

In higher dimensions, the first obstacle is that equation (4.2) no longer gives us rational solutions. Indeed, looking at the two-dimensional case gives us

$$F((x,y),P) = -\frac{n+1}{2}(1-x^2)(1-y^2) + (1-x)(1-y) + 2\sum_{x_i \in P}(1-\max(x_{i,1},x))(1-\max(x_{i,2},y)).$$

$F((x,y),P)$ will be shortened to $F(y,P)$ for readability when the context is clear, with y a vector. As in the one-dimensional case, we can consider the boxes of the type $[x_{i,1},x_{j,1}]\times[x_{k,2},x_{l,2}]$ such that all the maxima above take a fixed value (either the constant coordinate, or one of the parameters of y). We will call such boxes *gridboxes*.[58] These gridboxes form a grid of size $(n+1)\times(n+1)$, each gridbox will be referenced by a pair of integers (i,j). For example, $(n+1,n+1)$ corresponds

58 They correspond exactly to the boxes defined by the elements of $\overline{\Gamma(P)}$.

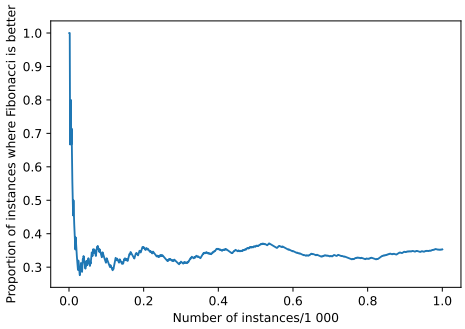

Figure 8.6: Proportion of n for which the Fibonacci sequence is better than the Kritzinger one, initialized in $x = 0.5$, counting only one instance per 1000 points. The Kritzinger sequence appears to perform better in 60% to 70% of the cases, and this appears to be relatively stable.

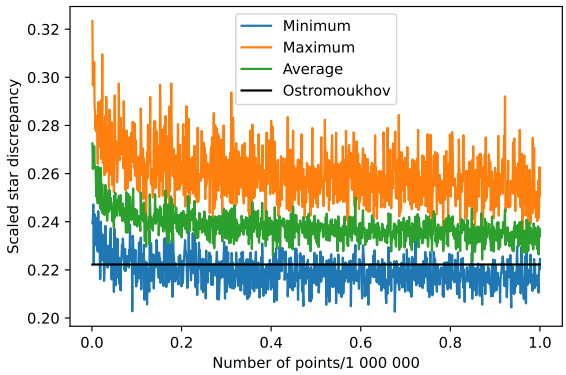

Figure 8.7: Best, average and worst scaled discrepancy out of 11 sequences with different single starting points for the Kritzinger sequence in dimension 1. The excellent performance of the maximum of the sequences shows the robustness of the Kritzinger sequence.

to the top-right box. Inside each gridbox, we want to find the minimum of the following 2-variable fourth-order polynomial $F_{i,j}$.

$$F_{i,j}((x,y),P) := -\frac{n+1}{2}(1-x^2)(1-y^2) + (1-x)(1-y) + A_{i,j}(1-x)(1-y) + B_{i,j}(1-x) + C_{i,j}(1-y) + D_{i,j} \tag{8.3}$$

$A_{i,j}$, $B_{i,j}$, $C_{i,j}$ and $D_{i,j}$ are constants that only depend on the choice of the gridbox.

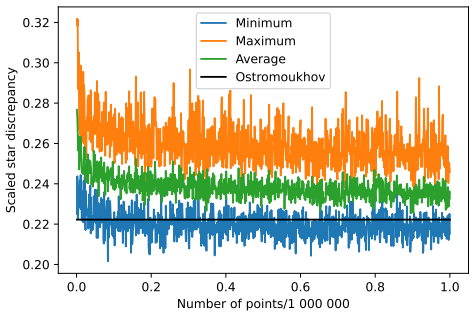

Figure 8.8: Best, average and worst scaled discrepancies out of 6 sequences with 5 random initial points in dimension 1. Results are very similar to the one point case, further underlining the lack of dependency on the initial point of the Kritzinger sequence.

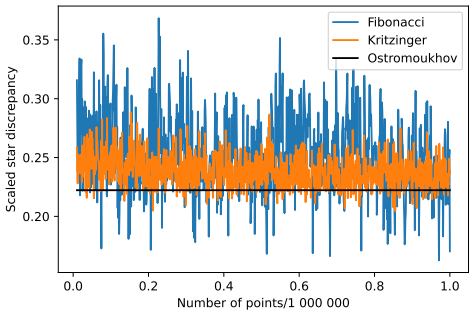

Figure 8.9: Scaled discrepancy values with a bad initial set of 100 points, for $n = 10\,000$ to one million.

If we want to find (x, y) such that $\nabla F_{i,j}((x, y), P) = 0$, we need to find (x, y) solution of the two following equations.

$$(n+1)(1-y^2)x - (A_{i,j}+1)(1-y) - B_{i,j} = 0 \tag{8.4a}$$
$$(n+1)(1-x^2)y - (A_{i,j}+1)(1-x) - C_{i,j} = 0 \tag{8.4b}$$

Expressing x as a function of y in equation (8.4a), reinserting it in equation (8.4b), and eliminating the fractions gives us a *ninth*-order polynomial. This problem cannot be tackled by off-the-shelf solvers and other methods need to be found. We describe some possibilities in the next two sections: an exact method in Section 8.7.1.1 and approximate approaches in Section 8.7.1.2. Both give relatively

comparable results, which will be described in Section 8.7.2. The methods will be described for two dimensions but can easily be generalized to higher dimensions.

8.7.1.1 Exact Construction

This section is inspired by the methods in [Clé+23a] presented in Chapter 5 to optimally place points for the L_∞ star discrepancy. We can express our problem of optimizing the placement of a point in an existing set with regards to the L_2 star discrepancy as a non-linear mathematical programming problem.

$$\min \quad -\frac{n+1}{2}v_3 + (1-y_1)(1-y_2) + 2\sum_{j=1}^{n} t_j u_j$$

$$
\begin{aligned}
& x_{2j-1} = x_{j,1} && \forall j = 1,\dots,n && \text{(8.5a)}\\
& x_{2j} = x_{j,2} && \forall j = 1,\dots,n && \text{(8.5b)}\\
& r_j - 1 \le x_{2j-1} - y_1 && \forall j = 1,\dots,n && \text{(8.5c)}\\
& r_j \ge x_{2j-1} - y_1 && \forall j = 1,\dots,n && \text{(8.5d)}\\
& s_j - 1 \le x_{2j} - y_2 && \forall j = 1,\dots,n && \text{(8.5e)}\\
& s_j \ge x_{2j} - y_2 && \forall j = 1,\dots,n && \text{(8.5f)}\\
& t_j = (1-r_j)y_1 + r_j x_{2j+1} && \forall j = 1,\dots,n && \text{(8.5g)}\\
& u_j = (1-s_j)y_2 + s_j x_{2j+2} && \forall j = 1,\dots,n && \text{(8.5h)}\\
& v_1 = y_1 y_1 && && \text{(8.5i)}\\
& v_2 = y_2 y_2 && && \text{(8.5j)}\\
& v_3 = (1-v_1)(1-v_2) && && \text{(8.5k)}\\
& r_j, s_j \in \{0,1\} && \forall j = 1,\dots,n, &&\\
& t_j, u_j \in [0,1] && \forall j = 1,\dots,n, &&\\
& x_j \in [0,1] && \forall j = 1,\dots,2n, &&\\
& y_1, y_2, v_1, v_2, v_3 \in [0,1] && && &
\end{aligned}
$$

Model (8.5) describes the objective and constraints used in the two-dimensional case, this can be easily generalized to higher dimensions. The objective corresponds to the minimization of $F((y_1, y_2), P)$, with some reformulations explained below. The constraints (8.5a) and (8.5b) fix the values of the previous points. Each of the maximum terms in the final sum needs to be reformulated for the solver. For this, we introduce binary variables in equations (8.5c) to (8.5f) (r_j for the first dimension and s_j for the second) that are equal to 1 if the constant is greater, and 0 if our

variable is bigger. For each of these binary variables, two constraints are required:

$$r_j - 1 \leq x_{2j-1} - y_1 \quad \forall j = 1, \ldots, n$$

$$r_j \geq x_{2j-1} - y_1 \quad \forall j = 1, \ldots, n$$

The first constraint imposes that if the constant coordinate is smaller, then r_j is equal to 0. The second fixes r_j to 1 if the constant coordinate is bigger. Variable t_j in constraint (8.5g) then plays the role of the maximum function for the first coordinate, while variable s_j in constraint (8.5h) plays a similar role for the second coordinate. While no other constraints are theoretically required, solvers such as Gurobi [Gur23] cannot handle products of more than two variables easily. Terms such as $y_1^2 y_2^2$ have to be replaced. For this, we introduce three extra variables v_1 to v_3 in constraints (8.5i) to (8.5k), simply to reformulate the objective.

With this method, we are able to solve problems in a reasonable time for up to a few hundred points. We will use the sets obtained here as a baseline to guarantee that our approximate methods are relatively trustworthy.

8.7.1.2 Approximate Methods

We tried three different heuristics: searching on a discrete grid, random search, and gradient descent. With these, we are able to compute a few thousand points.

The discrete grid: Searching on a discrete grid was done by changing our search space $[0, 1]$ to a grid $\{(k/m, l/m) : k, l \in \{0, \ldots, n-1\}\}$, where $1/m$ is the grid stepsize. Given our current point set P, for each point y on the grid, we evaluate $F(y, P)$ and simply choose the point that minimizes this function as our next point. As in the one-dimensional case, we can do this better than the naïve $O(nm^2)$ runtime by tracking which gridbox our search point is in. This means we only need to track the value of the constants $A_{i,j}$, $B_{i,j}$, $C_{i,j}$, and $D_{i,j}$ to evaluate $F_{i,j}(y, P)$ in constant time. There may be no or multiple points per gridbox, but this does not impact the overall $O(m^2)$ complexity.

However, this method had a drawback in that it could ignore some of the $F_{i,j}$ if m was too small. We therefore adapted it to defining a sub-grid *per gridbox and function* $F_{i,j}$. For each point on this sub-grid, we evaluate $F_{i,j}$ and keep the best performing point. This point is then compared to those obtained with the other $F_{i,j}$, and the best one is kept. This increases the global complexity: there are $O(n^2)$ functions $F_{i,j}$, and for each of these we evaluate $O(m^2)$ points, but it guarantees that we have considered the whole search space.

Random search: To avoid having to recompute $F(y, P)$ for every point, we

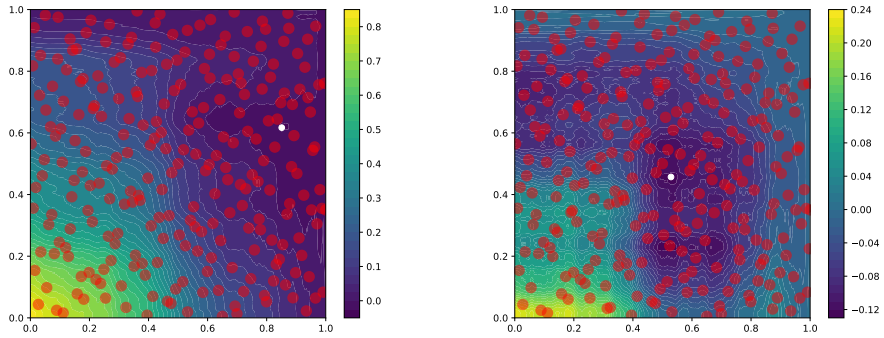

Figure 8.10: Plots of $F(y, P)$ for the Kritzinger sequence initialized in $(0.5, 0.5)$ with 247 (left) and 266 points (right). The point minimizing $F(y, P)$ is shown in white (respectively the 248-th and 267-th points of the sequence). $F(y, P)$ is very smooth, suggesting gradient descent methods can be used locally.

re-use the grid structure and the $F_{i,j}$ functions. For each $F_{i,j}$ function, we sample a fixed number of points s (around 100 million in 2 dimensions, 10 billion in 3 dimensions), scaled by the volume of the box. Not only should a larger box have more samples to guarantee a better exploration but, since we are trying to spread points out, a small box might indicate we are already close to existing points and thus less likely to pick a point there. The complexity to compute the next point with the random search method is in $O(sn^2)$.[59]

Gradient descent: While we cannot easily find an exact solution for our polynomials, they remain of a relatively low degree and we are searching for solutions in a relatively small space (initially $[0, 1]^2$, then smaller sub-boxes). Numerical experiments to plot $F(y, P)$ over $[0, 1]$ for around 250 points are shown in Figure 8.10. We see that for the Kritzinger sequence $F(y, P)$ is relatively smooth and, while there are multiple local minima, they correspond to different $F_{i,j}$. We can therefore hope that inside each gridbox (i, j), there is a single minimum and that we can do gradient descent to find $\arg\min_{(x,y)} F_{ij}((x, y), P)$. This is the most expensive method as it requires running a gradient descent algorithm inside $O(n^2)$ boxes to determine the $(n + 1)$-th point.

As shown in the next section, the three methods provide sequences of similar quality. We note that gradient descent seems to be able to obtain nearly the same

59 This could be improved with the use of a low-discrepancy set to sample in the desired zones when n becomes big. For low n, we sample so many points that it has relatively little importance.

points as the Kritzinger sequence when n is low in two dimensions, but is also the worst performing method once n and d increase. The grid method appears to be inferior to the random approach in all cases.

8.7.2 Results in Dimensions 2 and 3

In this section, we present our results in dimensions 2 and 3. While the methods are valid for any dimension, they get more expensive as the dimension increases. Furthermore, there is a great instability in the point choices. In $1d$, picking a point slightly off would have little impact for the next few points, as we know that they respect a specific structure. This is no longer the case in higher dimensions: choosing one point very slightly differently can lead to instant changes for the next point and then lead to very different sequences. We initialized 10 different sequences with $(0.5, 0.5)$ and used the random heuristic on them separately to obtain 500 points. While they all had very similar discrepancy values over the 500 points, the sequences generally became very different around the 70 points mark. In some cases, two of the sequences were identical up to 70 points and did not have a single point in common after the 80th. As the dimension increases, this leads to much greater mistakes in the sequence.

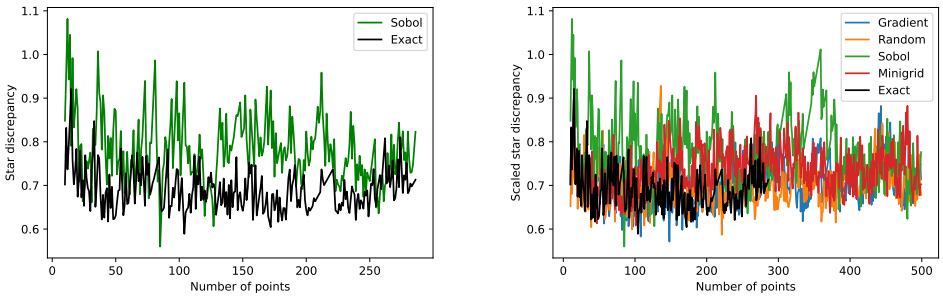

Figure 8.11: On the left, comparison of the exact Kritzinger sequence initialized in (0.5,0.5) compared with the Sobol' sequence. On the right, comparison of the different methods initialized in (0.5,0.5) with the Sobol' sequence. In both, discrepancies are scaled by $n/\log(n)$.

Nevertheless, our results show that the quality of the sequence remains robust, at least in 2 dimensions. In particular, Figure 8.11 (left) gives the L_∞ star discrepancy of the first 277 points of the Kritzinger sequence initialized in $(0.5, 0.5)$, compared to the Sobol' sequence. This is scaled by $n/\log(n)$ (not squared, otherwise the plot would have needed to be truncated for readability). Figure 8.11 (right) then compares the approximate methods with the exact sequence and the Sobol' sequence

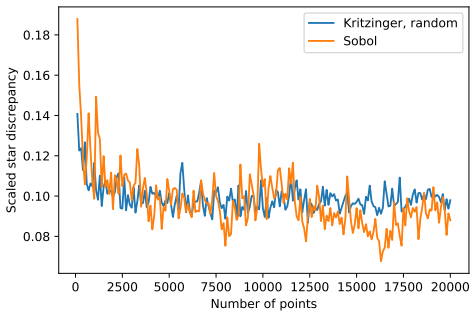

Figure 8.12: Random heuristic and Sobol' sequence L_∞ star discrepancies for up to 20 000 points in dimension 2. L_∞ star discrepancy values are scaled by $n/\log^2(n)$.

in the same context. We see that our approximate methods produce sequences that are quite similar to the exact sequence, at least for the discrepancy values obtained. All of them are competitive with the Sobol' sequence for all values shown. This suggests to some degree that like in the one-dimensional experiments with different starts, greedy L_2 sequence construction is not too dependent on the exact points themselves for good performance. Even when increasing n further as in Figure 8.12, we observe that the random method generates a sequence whose L_∞ star discrepancy values are comparable to those of the Sobol' sequence over the first 13 000 points. While there seems to be a big difference at 16 000 points, one should remember that the construction method of the Sobol' sequence leads to better values when $n = 2^k$ for $k \in \mathbb{N}$. Combining this with the low precision of the method partly explains the discrepancy gap.

Figure 8.13 describes our results in dimension 3. The exact Kritzinger sequence consistently outperforms the Sobol' sequence, while the random approach seems to perform decently. Both the gradient and discrete grid approaches are a lot less successful. We note that while we are able to compute more points with the random approach, their quality degrades around $n = 500$. Figure 8.13 shows that our random points approach is generally outperformed by the Sobol' sequence. Nevertheless, the discrepancy order seems to still be $O(\log(n)^3/n)$, despite the imprecisions in the construction.

Our empirical results in 2 and 3 dimensions, suggesting quite strongly that the discrepancy is of the same order as for the Sobol' sequence, lead us to formulate the following conjecture.

▶ **Conjecture 8.2.** The Kritzinger sequence has a discrepancy of $O(\log^d(n)/n)$ in dimension d. ◀

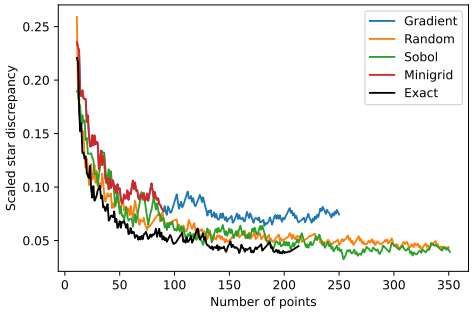

Figure 8.13: Comparison of our different methods in dimension 3 with the Sobol' sequence. Values are scaled by $n/\log^3(n)$.

8.8 Conclusion

This section introduced or developed greedy construction methods in low dimensions, with the aim of obtaining low L_∞ star discrepancy sequences. While the results for the greedy L_∞ method are poor, they provide an answer to a relatively common question (i.e. how to add a point optimally to a set), but more importantly seem to open the possibility of doubling the number of points inside a set. Indeed, model (8.2) can easily be extended to add any number of points, and not just one. Preliminary experiments show that very good sets of size $2n$ can be obtained by starting with a good n point set and optimizing the placement of another n. While this project is only in its infancy, such decomposition methods could provide useful in all contexts.

For the Krizinger sequence, we provided a set of methods to compute it, both to greatly extend the number of points in one dimension and to give the first results in higher dimensions. Our exact and heuristic approaches seem to confirm first observations from [Kri22], in that the Kritzinger sequence appears to be a low-discrepancy sequence. In particular, our one-dimensional results suggest it could be even better than some of the best sequences known. For higher dimensions, further work will be required to make heuristics more precise and prevent the deviations we observe when n increases. Indeed, while exact optimization of $F(y, P)$ seems to outperform the Sobol' sequence, heuristic approximations are "only" comparable. Identifying feasible methods of solving the polynomials to obtain exact minima of the $F_{i,j}(y, P)$ could be an interesting direction.

Nevertheless, the sequences resulting from these deviations are still excellent. This suggests that the greedy L_2 discrepancy construction is robust: given a starting set, this construction method will provide a good low-discrepancy sequence rela-

tively quickly. In particular, it should be possible to "fix" any unbalanced set with greedy point additions, as we showed in one dimension. Both the robustness and the correction power, that one can naturally associate with *sequences*, should make such methods based on the L_2 discrepancy more important for L_∞ constructions in the future.

9 Black-Box Optimizers for L_∞ Star Discrepancy Computation

This chapter and the next present a different focus than the rest of the thesis, in that rather than trying to construct high quality low L_∞ star discrepancy sets, we will try to highlight potential uses of black-box optimizers for discrepancy-related problems. This chapter corresponds to [Clé+23c] and it is joint work with Diederick Vermetten, Jacob de Nobel, Alexandre D. Jesus, Carola Doerr and Luís Paquete. In particular, Alexandre D. Jesus was responsible for the implementation of the parallel DEM algorithm.

9.1 Summary of Results

This chapter describes the use of black-box optimizers to compute the L_∞ star discrepancy of a given set. As described in Chapter 1, point sets of low discrepancy are required for a multitude of applications. Some of these require discrepancy calculations in much higher dimensions, in some cases $d \geq 100$, far out of reach of the exact algorithms presented in Chapter 3 for the moment. Despite the complexity of finding the global maximum, evaluating the local discrepancy in a given point q can be done very efficiently. Black-box optimization approaches can therefore be used to tackle this problem, as even making a very large number of local evaluations is feasible.[60]

We apply eight classical optimizers described in Section 9.3 on three different point set types, for varying dimensions and set sizes. We show in Section 9.4 that these default popular numerical black-box optimizers are unsuccessful in computing the L_∞ star discrepancy, even for instances that can be solved within a second with naïve methods. The relative performance rapidly decreases with increasing dimension, whereas the size of the point sets only seems to have a minor impact on the overall (bad) quality of the solvers, which might be surprising given that the differences in the local star discrepancy values at the discontinuities become smaller with increasing set size (inverse linear relationship). The sampler type used to generate the point sets has relatively little importance, if any, on the performance of the optimizers.

60 This is also what makes the Threshold Accepting algorithm in Section 3.4 work in any dimension.

We observe that uniform i.i.d. random search seems to be the best performing optimizer over the vast majority of the instances, yet still fails to come close to the exact value, as highlighted by a relative performance comparison with the known exact values. We suspect that state-of-the-art numerical black-box optimization techniques fail to capture the global structure of the problem, an important shortcoming that may guide their future development.

To have a better point of comparison in moderate dimensions, we first provide a new parallel implementation of the DEM algorithm [DEM96] in Section 9.2. We show that our parallel implementation has a speed-up factor of up to 17 on 32 threads, while enabling better evaluations of the quality of the results obtained by the optimizers.

Finally, our implementations can be found in a Zenodo repository [Clé+23b], along with past implementations of the DEM algorithm and TA heuristic by the authors of [GWW12]. The repository also contains all code and data used for the analysis of the black-box optimization algorithms, as well as the code used to process this and create the figures included in [Clé+23c].

9.2 Parallelizing DEM

We recall that the original DEM algorithm was described in Section 3.3, and refer the reader to this section for details on the algorithm.

To parallelize the DEM algorithm, we start by noting that after computing the boxes for a given dimension we can continue the recursive decomposition independently for each box. This naturally gives way to a parallel-task construct where each task that can be scheduled to an available CPU corresponds to the decomposition in dimension $j+1$ of a given box B_j.

At this point, it is worth noting that another option initially considered was to parallelize the dynamic programming algorithm that is used to compute the worst local discrepancy for a box after d steps. However, this idea was discarded since the dynamic programming algorithm was very fast in practice and preliminary attempts did not yield significant speedups.

To implement the parallel-tasks construct we considered OpenMP Tasks [Ayg+09] and we adapted the implementation of the DEM algorithm by the authors of [GWW12]. In particular, one relevant change was that the sequential implementation reused the set of points between recursive calls, with a possible resorting of the points in the current dimension after each recursive call. However, in the parallel variant, we cannot efficiently share the set of points since sorting operations on the set of points in a thread could potentially affect tasks

running in other threads. Instead, in the parallel DEM we opted to copy the set of points before entering a recursive call. Although this requires an extra copy before each recursive call, we observed that in some cases copying the vector was faster than resorting after each recursive call, suggesting that the non-parallel DEM implementation could be slightly improved by taking this into account.

Finally, in order to avoid thread starvation with a recursive parallel-tasks construct, it is often useful to consider a cut-off value related to the problem size to switch to a non-parallel variant. In this case, we define the cut-off with respect to the complexity of the DEM algorithm, i.e., $O(n^{1+d/2})$. In particular, when decomposing a box B_j with n_j points in dimension j, we switch to the non-parallel DEM algorithm if ${n_j}^{1+(d-j)/2} < 1e8$. This cut-off value was defined empirically by experimenting with different values up to $1e15$.

Table 9.1 gives the CPU time in seconds for the parallelized DEM on several instances generated with the GNU Scientific Library (using the indicated sequence) and considering different numbers of threads. For a single thread, the best time between the non-parallel and parallel implementations is shown. Experiments were run on a machine with two Intel(R) Xeon(R) Silver 4210R CPUs clocked at 2.40GHz, comprising a total of 20 cores and 40 threads. We can see that for 32 threads we have speedups that range from a factor of 7.4 to a factor of 17.4. These results show that parallelization is generally successful and can give a significant boost. However, we believe that there is still room for improvement by further tuning the cut-off between the parallel and non-parallel variants, and by further analyzing when it is best to copy or resort the set of points.

Table 9.1: CPU time in seconds and corresponding speedup in parenthesis for the parallelized DEM algorithm

			CPU time (speedup) per number of threads					
d	n	Sequence	1	2	4	8	16	32
2	**50 000**	Halton	2.54 (1.0)	1.56 (1.6)	0.88 (2.9)	0.53 (4.8)	0.38 (6.7)	0.32 (7.9)
		Niederreiter	2.50 (1.0)	1.56 (1.6)	0.87 (2.9)	0.54 (4.6)	0.34 (7.4)	0.31 (8.1)
		Reversehalton	2.37 (1.0)	1.59 (1.5)	0.86 (2.8)	0.53 (4.5)	0.40 (5.9)	0.32 (7.4)
		Sobol	2.39 (1.0)	1.58 (1.5)	0.86 (2.8)	0.53 (4.5)	0.41 (5.8)	0.31 (7.7)
3	**10 000**	Halton	15.70 (1.0)	8.19 (1.9)	4.31 (3.6)	2.37 (6.6)	1.52 (10.3)	1.13 (13.9)
		Niederreiter	15.77 (1.0)	8.09 (1.9)	4.36 (3.6)	2.39 (6.6)	1.65 (9.6)	1.16 (13.6)
		Reversehalton	15.57 (1.0)	7.99 (1.9)	4.37 (3.6)	2.37 (6.6)	1.54 (10.1)	1.17 (13.3)
		Sobol	15.74 (1.0)	8.19 (1.9)	4.38 (3.6)	2.39 (6.6)	1.44 (10.9)	1.14 (13.8)
4	**3 000**	Halton	49.66 (1.0)	26.40 (1.9)	13.97 (3.6)	7.38 (6.7)	3.99 (12.4)	2.89 (17.2)
		Niederreiter	50.31 (1.0)	26.82 (1.9)	14.33 (3.5)	7.42 (6.8)	3.96 (12.7)	2.89 (17.4)
		Reversehalton	50.42 (1.0)	26.53 (1.9)	14.11 (3.6)	7.44 (6.8)	4.05 (12.4)	2.96 (17.0)
		Sobol	50.41 (1.0)	27.20 (1.9)	14.31 (3.5)	7.32 (6.9)	4.06 (12.4)	2.92 (17.3)
5	**1 000**	Halton	58.87 (1.0)	32.54 (1.8)	16.70 (3.5)	9.18 (6.4)	5.10 (11.5)	3.83 (15.4)
		Niederreiter	62.32 (1.0)	33.39 (1.9)	17.78 (3.5)	9.67 (6.4)	5.34 (11.7)	4.08 (15.3)
		Reversehalton	62.08 (1.0)	32.77 (1.9)	17.54 (3.5)	9.64 (6.4)	6.88 (9.0)	3.90 (15.9)
		Sobol	63.80 (1.0)	33.89 (1.9)	17.73 (3.6)	9.77 (6.5)	5.34 (11.9)	4.04 (15.8)
6	**600**	Halton	175.79 (1.0)	103.74 (1.7)	53.04 (3.3)	28.82 (6.1)	17.12 (10.3)	12.57 (14.0)
		Niederreiter	196.24 (1.0)	109.22 (1.8)	57.99 (3.4)	30.44 (6.4)	17.94 (10.9)	12.71 (15.4)
		Reversehalton	187.99 (1.0)	114.35 (1.6)	58.75 (3.2)	31.56 (6.0)	19.12 (9.8)	13.49 (13.9)
		Sobol	200.76 (1.0)	112.53 (1.8)	59.18 (3.4)	30.91 (6.5)	18.34 (10.9)	13.32 (15.1)
7	**300**	Halton	160.25 (1.0)	94.43 (1.7)	48.22 (3.3)	27.10 (5.9)	15.64 (10.2)	11.70 (13.7)
		Niederreiter	167.32 (1.0)	96.85 (1.7)	51.36 (3.3)	28.40 (5.9)	15.80 (10.6)	12.66 (13.2)
		Reversehalton	149.43 (1.0)	88.79 (1.7)	50.57 (3.0)	29.72 (5.0)	16.63 (9.0)	12.49 (12.0)
		Sobol	169.70 (1.0)	102.12 (1.7)	52.72 (3.2)	29.68 (5.7)	16.20 (10.5)	12.41 (13.7)

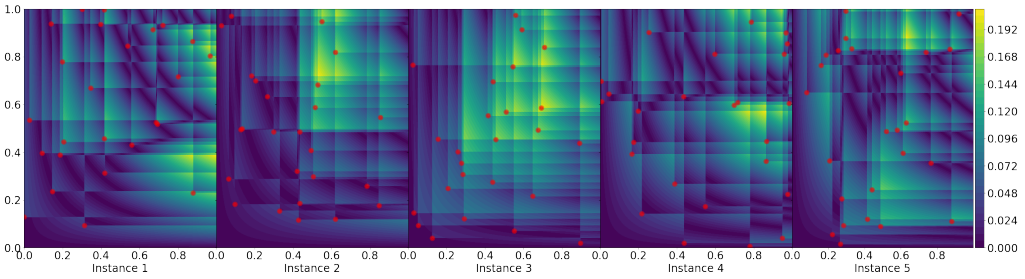

Figure 9.1: Local discrepancy values of the first 5 instances of the 2-dimensional version of F31: Uniform sampler with 20 points. The red points indicate the originally sampled points.

9.3 Numerical Black-Box Optimization Approaches

Figure 9.1 illustrates the local L_∞ star discrepancy values for an instance in $d = 2$. We highlight two properties of the L_∞ star discrepancy function which, while probably clear to the reader who has reached this chapter, we prefer to underline.

- It is a *multimodal* problem, i.e., there can be several local optima in which the solvers can get trapped (this problem becomes worse with increasing dimension);
- Discontinuous discrepancy values. Slightly increasing one coordinate can result in a point falling inside the considered box, causing a $1/|P|$ difference in the local star discrepancy value. Figure 9.2 shows that the problem structure also depends strongly on the point set considered.

The L_∞ star discrepancy problem was included as a black-box benchmark problem in IOHexperimenter (version 0.3.7) [Nob+24], using three different point set generators:

1. uniform random sampling,
2. Halton [Hal60],
3. Sobol' [Sob67].

For the uniform sampler, a standard Mersenne Twister 19937 pseudo-random number generator was used, provided by the C++ STL. The Halton sequence was generated using a classic Sieve of Eratosthenes prime number generating algorithm, and the Sobol' sequence was generated with a third-party library based on the FORTRAN implementation of [Fox86].

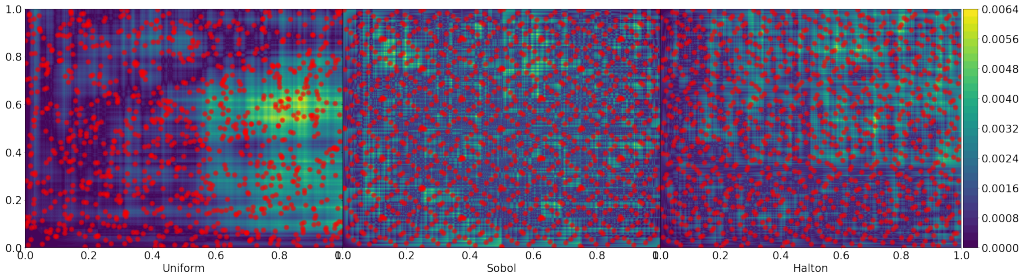

Figure 9.2: Comparison of the 3 samplers for 1000 points in 2D, and the corresponding discrepancy landscapes (instance 1 for F39, F49 and F59 respectively).

In addition to the generator, the number of points to be sampled can be selected, which along with the dimensionality, controls the complexity of the problem. We define a default suite, which includes for every generator a fixed number of samples $S \in \{10, 25, 50, 100, 150, 200, 250, 500, 750, 1000\}$. This includes a total of 30 benchmark problems, where for each problem the dimension and instance can be varied arbitrarily. Instances are controlled by a unique instance identifier, which is a positive integer that determines the random seed used to generate the point set. The problems can be accessed through both the Python and C++ interfaces of IOHexperimenter, with problem ids $\{30, \ldots, 39\}$, $\{40, \ldots, 49\}$, and $\{50, \ldots, 59\}$ for the problems generated with the uniform, Sobol', and Halton generators, respectively.

We test a total number of eight algorithms, all taken from the Nevergrad platform [RT18]:[61]

1. Diagonal Covariance Matrix Adaptation Evolution Strategy (dCMA-ES) [HO01]
2. NGOpt14, Nevergrad's algorithm selection wizard [Meu+21]
3. Estimation of Multivariate Normal Algorithm (EMNA) [LL01]
4. Differential Evolution [SP97]
5. Constrained Optimization BY Linear Approximation (Cobyla) [Pow94]
6. Random Search

61 See Section 10.3 for a short description of the CMA-ES and NGOpt optimizers.

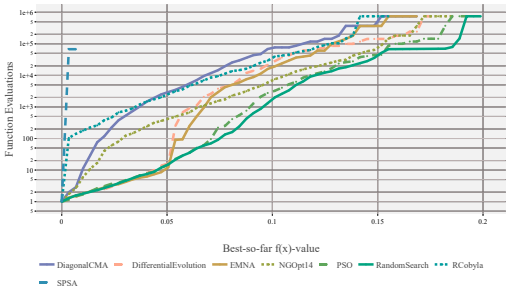

Figure 9.3: Expected running time (ERT) of the 8 optimization algorithms on the discrepancy calculation for the uniform sampler with 100 samples (F33) in 3 dimensions. ERT is calculated based on 10 runs on 10 instances with a budget of 7 500. Figure generated using IOHanalyzer [Wan+22].

7. Particle Swarm Optimization (PSO) [KE95]

8. Simultaneous Perturbation Stochastic Approximation algorithm (SPSA) [Spa92].

The algorithms are chosen "as they are" from Nevergrad. That is, we did not perform any hyper-parameter tuning nor did we change any of their components. Each algorithm is given a total budget of $2\,500 \cdot d$ local discrepancy evaluations, where as always d is the dimensionality of the problem. For each instance, 10 independent runs of the algorithm are performed. This is repeated for 10 instances, resulting in a total of 100 runs per function, for each of the 30 functions, with dimensionality $d \in \{2, 3, 4, 6, 8, 10, 15\}$.

We run all our experiments in the IOHprofiler environment [Doe+18]. This allows us to track not only the final performance, but also the trajectory of the algorithms in the objective space. It furthermore allows a straightforward visualization and analysis of the data using the IOHanalyzer module [Wan+22].

For all instances in dimensions 2, 3 and 4, and for all the instances in dimensions 6, 8, 10, and 15 with n not larger than 750, 200, 50 and 10 points respectively, we computed the exact discrepancy values using the parallel DEM algorithm proposed in Section 9.2. For all other instances, we computed a lower bound for the star discrepancy value using the TA algorithm described in Section 3.4.

9.4 Results

We can compare the performance of the used optimization algorithms by considering the expected running time (ERT) to reach increasing discrepancy values. This analysis shows the convergence behavior on the selected function. In Figure 9.3, we show the ERT on the 3-dimensional version of F33: the uniform sampler with $n = 100$. From this figure, we can clearly see that the algorithms struggle to optimize this function. Particularly noticeable is the SPSA algorithm, which seems to fail to find even slightly improved discrepancy values. On the other side, we notice that Random Search is surprisingly outperforming all other optimizers. This seems to indicate that even for this relatively simple setting of $n = 100$ and $d = 3$, the high level of multi-modality combined with discontinuities in the landscape cause problems for all of the considered optimization algorithms.

To check whether this is a consistent problem, or something specific to the settings chosen in Figure 9.3, we can look in more detail at the final solutions found by each optimizer across a wider set of scenarios. In order to create a fair comparison, we can move from the original discrepancy values to a relative measure, based on the bounds found by the TA and DEM algorithms. Specifically, we consider the following measure:

$$R(x) = \frac{OPT(x) - f(x)}{OPT(x)},$$

where $f(x)$ is the final value found after the optimization run, and $OPT(x)$ is the bound calculated by the parallel DEM or TA algorithms, depending on the instance size.

Using this relative measure, we can compare the final solutions found by each optimization algorithm across a set of different n and d. This is visualized in Figure 9.4. In this figure, we see that the observations made based on ERT from Figure 9.3 seem to hold across scenarios: the SPSA algorithm is clearly performing poorly, while Random Search seems to be competitive with, if not superior to, all other algorithms for every scenario. In addition to the ranking between algorithms, we also note a clear increase in problem difficulty as the dimensionality increases. Conversely, the number of samples seems to have a rather limited impact on the relative difficulty. This suggests that the structure of the point set has little influence on the performance of the optimizers.

As a final comparison, we can consider the differences in the difficulty of the optimization problem when different samplers are used. As we observed in Figure 9.2, the landscape is clearly impacted by the choice of the sampler. To see whether this also impacts the performance of the optimization algorithms, we consider

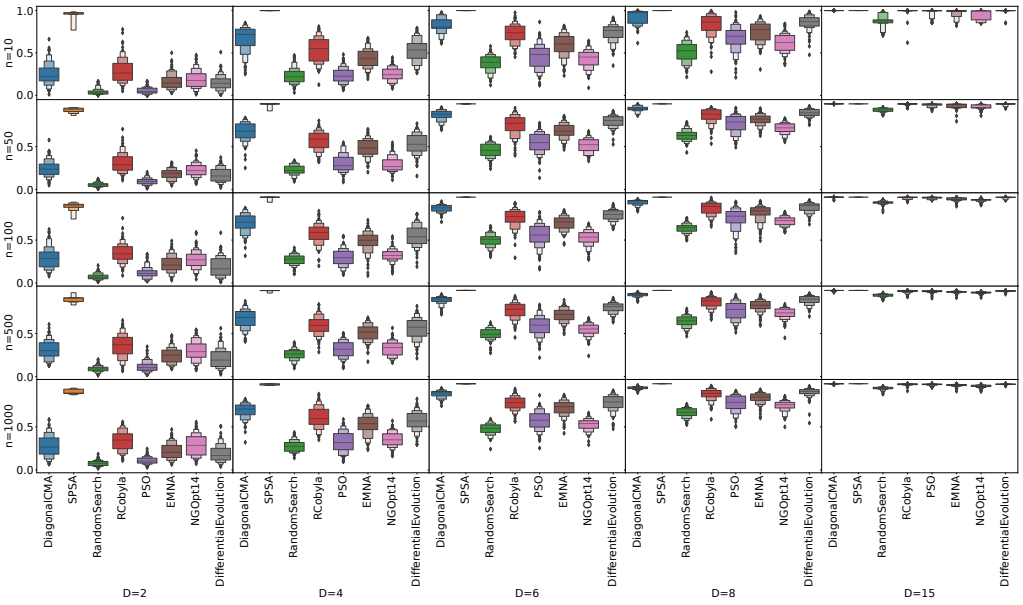

Figure 9.4: Relative final discrepancy value found by each of the used optimizers. Values of 0 correspond to finding the optimal solution, while 1 corresponds to the worst achievable value (the solver believes the set has a discrepancy of 0). Box-plots are aggregations of 10 runs on 10 instances, all for the uniform sampler.

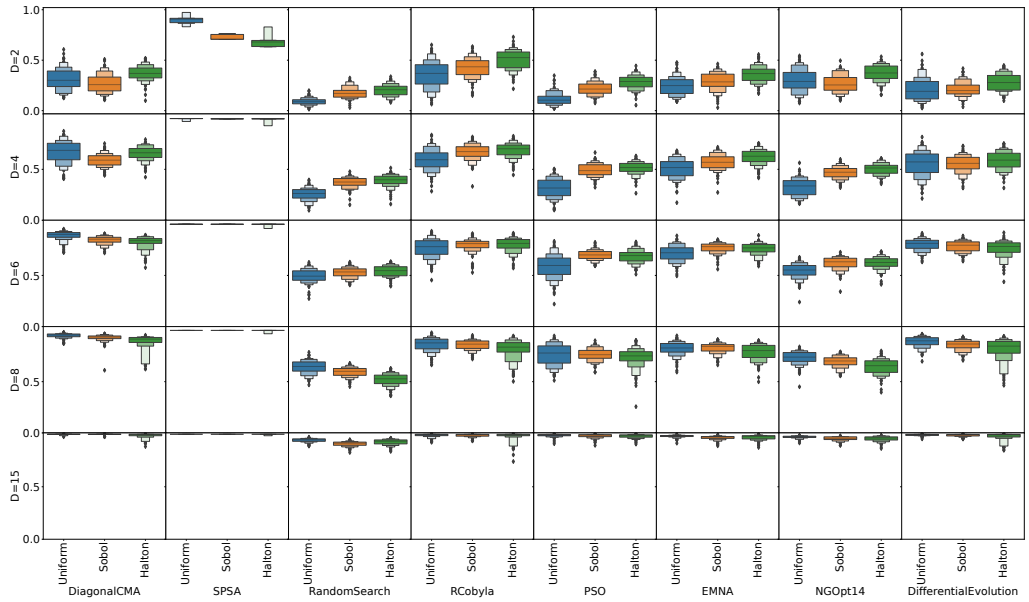

Figure 9.5: Relative final discrepancy value found by each of the used optimizers, compared between different samplers. Values of 0 correspond to finding the optimal solution, while 1 corresponds to the worst achievable value (the solver believes the set has a discrepancy of 0). Boxplots are aggregations of 10 runs on 10 instances, all for $n = 500$.

the relative final discrepancy found on the grids with $n = 500$, for a few selected dimensions. The resulting distributions are visualized in Figure 9.5. From this figure, we can see that, while the choice of sampler often has a low impact on the performance of most algorithms, some tendencies can still be observed. In most cases, the Halton sampler seems to be the most challenging, especially in lower dimensions. However, the ordering is not fully consistent between algorithms or even between dimensions.

9.5 Conclusions

We have studied the efficiency of numerical black-box optimization approaches for maximizing the local L_∞ star discrepancy values for a given point set P. The results are underwhelming; the obtained results cannot be used as reliable estimates for the overall L_∞ star discrepancy of P.

The results indicate that off-the-shelf black-box optimization approaches have difficulties coping with the multi-modal nature of the problem and/or with the dis-

continuities in the genotype-phenotype mapping. We believe that this combination makes the problem an interesting use case for comparing diversity mechanisms with or without restarts. In particular, we expect that approaches such as *quality-diversity* (originally introduced in [PSS16; Pug+15], but see [Cha+20] for a more recent survey) or *niching* [Shi12] could improve the quality of the search algorithms.

The focus of our work has been on the numerical black-box solvers. However, an interesting aspect of the star discrepancy computation problem is that it can also be studied as a discrete problem, using the grid structure described in Section 3.1. This grid structure is exploited by the TA algorithm from [GWW12] (see Section 3.4). We suspect that merging some of the problem-specific components of this algorithm (e.g., the *snapping* rounding routines) into evolutionary algorithms operating on discrete search spaces of the type $[1..n]^d$ could be worthwhile.

More importantly, the multimodularity of the problem seems to be a major hindrance to the performance of the tested black-box optimizers. Recent developments[62] suggest a better choice of optimizers lead to much better results. Indeed, Olivier Teytaud was able to compute exact values for all tried instances[63] in dimensions lower than 8. To our knowledge, this was based on a modified version of the NGOpt optimizer (see Section 10.3), tailored to multimodal problems. A deeper look into this could provide excellent results and a better alternative to the TA heuristic.

62 This is based on work by, and discussions with, Olivier Teytaud, posterior to the publication of the paper this chapter is based on.

63 These were taken from a GECCO competition organised in relation with our submission.

10 Black-box Optimizers for Stratified Sampling Optimization

This final chapter is a truncated version of [CKP24], where only the part relevant to our work is described. It is joint work with Nathan Kirk and Florian Pausinger.

10.1 Summary of Results

We describe in this short chapter our use of some common black-box optimizers to compute optimal stratifications with respect to the expected L_2 discrepancy. Section 10.2 recalls the definition of jittered sampling, the difference with stratified sampling and, more importantly, describes both the key questions preluding this work and the equivolume constructions with strata orthogonal to the diagonal suggested by Kirk and Pausinger in a preliminary version of [CKP24].[64] We will only list the major results that are relevant for our optimization comparison in Section 10.4. A reader interested in the mathematical proofs and methods involved in the constructions can find these in the original paper [CKP24].

Section 10.3 describes the three optimizers used, before Section 10.4 presents a comparison of the stratifications returned by the different optimizers, those obtained mathematically and known constructions with jittered sampling. Our optimization does not cover all possible stratifications, only all stratifications where each stratum is orthogonal to the main diagonal. We show that all three optimizers perform similarly, and the resulting kernel density plots suggest the optimal stratifications should not be equivolume, but alternating between clusters of thin strata and a large stratum. One of the limits during this optimization was the noise on the expected L_2 discrepancy value: the verified value with the returned set is always higher than the value it obtained during the run.

Even with this caveat, we obtain better results than the equivolume partitions on all tested instances in dimensions 2 and 3, for $n \leq 20$. We observe that, in two dimensions, the equivolume stratification $\mathcal{S}(N, 2)$ improves the uniformly distributed random point sets by a factor of 2 for the L_2 discrepancy, which is proven in [Pau23], and approximations of $\mathcal{S}^*(N, 2)$, for which we relax the equivolume condition, improve $\mathcal{S}(N, 2)$ by up to 8% for the smallest values of n. In

64 This is an essential part of [CKP24] but predates the optimization approach.

three dimensions, we see that approximations of $\mathcal{S}^*(N,3)$ provide a slightly larger improvement of up to 10% over the equivolume stratification.

10.2 Stratified Sampling

10.2.1 Jittered Sampling

Classical jittered sampling provides d-dimensional point sets with small discrepancy in $[0,1]^d$ consisting of $n = m^d$ points for a positive integer $m \geq 2$. As briefly mentioned in Section 2.2.1, the main idea is to partition the unit cube into n axis-parallel cubes of volume $1/m^d$ and to pick a uniform random point from each cube. This construction avoids local clusters one usually obtains when considering n i.i.d uniformly sampled points from $[0,1]^d$.

Equation (2.4) showing that jittered points have lower expected star discrepancy than random ones is our motivation to look for other constructions of partitions. In particular, we aim to find a construction that works for arbitrary n while improving the expected (star) discrepancy of a set of n i.i.d. uniform random points. In fact, for $1 < p < \infty$, a stratified set derived from a partition into $n > 2$ equivolume sets *always* has a smaller expected L_p-discrepancy than a set consisting of n i.i.d. random points. This *strong partition principle* was proven in [KP21, Theorem 1][65] and raises the question of which partition yields the stratified sample with the smallest mean L_p-discrepancy – if such a partition exists. Or, as a more modest goal, which construction works well in general?

10.2.2 An Equivolume Construction in Dimension 2

There are several straightforward ways to partition the d-dimensional unit cube $[0,1]^d$ into n sets of equal volume. We could for example partition the interval $[0,1]$ into n subintervals of equal length and use this to define n slices of equal volume. In fact, we can partition an arbitrary number of the d generating unit intervals into subintervals of equal length to generate grid-type partitions of the d-dimensional unit cube. In every such example it is straightforward to characterise the sets in the partition (since they are axis-parallel rectangles) and to prove that these sets have indeed all the same volume. However, any such construction only works for a restricted number of points and not for arbitrary n.

In this section, we characterise another family of equivolume partitions that was recently studied in the context of jittered sampling [KP21]. Given the unit square

65 See also [PS16] for a weaker form.

$[0,1]^2$ and $n-1$ parallel lines H_i with $i = 1, \ldots, n-1$, which are orthogonal to the main diagonal, D, of the square, we would like to arrange the lines such that we obtain an equivolume partition of the unit square; see Figure 10.1.

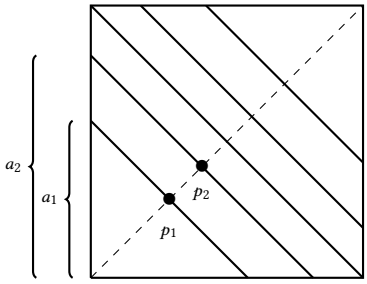

Figure 10.1: Partition of the unit square into $n = 6$ equivolume slices that are orthogonal to the diagonal.

We denote the intersection $H_i \cap D$ of a line with the diagonal with p_i. It is straightforward to calculate all points p_i for arbitrary n. In fact, note that H_i splits the unit square into two sets of volume i/n and $1 - i/n$. If $i \leq n/2$, we just need to look at the isosceles right triangle that H_i forms with $(0,0)$. We know that this triangle has volume i/n. Denote the length of the two equal sides with $a_i > 0$, then $a_i^2/2 = i/n$ and $p_i = a_i/\sqrt{2}$. Therefore, we get that

$$p_i = \sqrt{\frac{i}{n}}, \tag{10.1}$$

for all $i \leq n/2$. By symmetry, we also get the points p_i with $i > n/2$. The question behind the part of the paper not presented here is how to generalise this simple characterisation of equivolume partitions of the unit square to dimensions $d > 2$.

To state this problem formally, we introduce further notation. Let H_r^+ be the positive half space defined as the set of all $\mathbf{x} \in \mathbb{R}^d$ satisfying

$$x_1 + x_2 + \ldots + x_d \geq r; \tag{10.2}$$

accordingly let H_r^- be the corresponding negative half space and let H_r be the hyperplane of all points $\mathbf{x} \in \mathbb{R}^d$ satisying $x_1 + x_2 + \ldots + x_d = r$. For a given n, we would like to find the values $0 < r_1 < \ldots < r_{n-1} < d$ such that the corresponding hyperplanes $H_{r_1}, \ldots, H_{r_{n-1}}$ define a partition of $[0,1]^d$ into n equivolume sets. We call the set

$$\mathcal{S}(n,d) := \{r_i : i = 1, \ldots, n-1\}$$

the *generating set* of the partition. Using Equation (10.1) we see that

$$\mathcal{S}(n,2) = \left\{\sqrt{\frac{2i}{n}} : 1 \le i \le /2\right\} \cup \left\{2 - \sqrt{\frac{2i}{n}} : 1 \le i < n/2\right\}.$$

Note that for even n the point $\{1\}$ is in the set, while it is not for odd n.

Problem 1. *For given dimension d and number of sets n, characterise the set $\mathcal{S}(n,d)$.*

10.2.3 Widening the Stratification Search Space

Our approach is motivated by [KP21, Examples 2 and 3]. It was shown that among all partitions of the unit square into two sets of equal-volume, the dividing hyperplane orthogonal to the diagonal and going through the center of the square gives the smallest discrepancy. Furthermore, it was observed that moving the dividing hyperplane along the diagonal and thus relaxing the equivolume constraint can further improve the expected discrepancy; see Figure 10.2. This motivates our second question. For a given n, we would like to find the values $0 < r_1 < \ldots < r_{n-1} < d$ such that the corresponding hyperplanes $H_{r_1}, \ldots, H_{r_{n-1}}$ define a partition of $[0,1]^d$ into n sets minimizing the expected discrepancy of the corresponding stratified sample. We call the set

$$\mathcal{S}^*(n,d) := \{r_i : i = 1, \ldots, n-1\}$$

the *minimal set* of the parameter pair (n,d).

Problem 2. *For given dimension d and number of sets n, determine or approximate the minimal set $\mathcal{S}^*(n,d)$.*

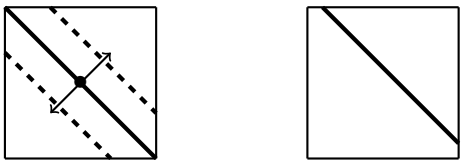

Figure 10.2: Left: Best partition into two sets of equal volume and illustration of one-parameter family of partitions obtained from moving hyperplane along diagonal. Right: The partition of this family with the smallest expected discrepancy.

While Problem 1 was the precursor of the work, Problem 2 will be the one interesting us in this chapter.

Without going into details[66], it was shown that:

$$\mathcal{S}(n,d) \approx \left\{ \frac{d}{2} + \frac{\sqrt{d}}{2\sqrt{3}} \Phi^{-1}\left(\frac{i}{n}\right) : i = 1, \ldots, n-1 \right\}. \tag{10.3}$$

We now tackle the problem of verifying if the partitions given by this formula match up against excellent, ideally optimal, partitions without the equivolume condition.

10.3 Three Common Optimizers

We quickly introduce in this section the three optimizers that will be used for our experiments, CMA-ES, NGOpt and a (1+1)-Evolutionary Strategy.

Black-box optimizers present two advantages for our problem. The first is that it is a very difficult task to directly compute the expected L_2-discrepancy of a general stratification. Recent approaches have always studied square boxes aligned with the axes [KP22] or used numerical experiments [KP21]; note that finding the best positions for 3 points was already an involved process, see [KP21]. Black-box optimizers, in contrast, do not require us to obtain new results on the *structure* of the function but directly try to find good intersection points for the given setting through a systematic trial and error process.

The second advantage is that for each intersection point, we only need to keep a single parameter since we know the hyperplanes are orthogonal to the diagonal, regardless of the dimension d. We can model the problem as an $n-1$ dimensional optimization problem $(p_1, \ldots, p_{n-1})$, where for each element p_i we have the bounds $p_i \in [0, \sqrt{d}]$, as well as the ordering constraints on the different parameters, i.e. $p_1 \leq p_2 \leq \ldots \leq p_{n-1}$.

CMA-ES The Covariance Matrix Adaptation Evolution Strategy, also known as CMA-ES is a family of heuristic optimization techniques for numerical optimization. It was initially introduced in [HO96] (with an extension in [HO01]), but modifications have been suggested over the years, leading to many different variants (see for example [Rij+16] for some popular modifications). We give here a brief description of the general principle, more details can be found in a tutorial paper by Hansen [Han16] as well as in a more recent presentation by Akimoto and Hansen [AH22]. In our experiments, we use the Diagonal-CMA version, described in [AH20].

66 We refer the interested reader to the full paper [CKP24].

The general principle behind CMA-ES is to introduce a covariance matrix associated to a multivariate Gaussian distribution and to learn second order information on this distribution. The mean of this distribution represents the current best guess while the covariance matrix determines the shape of the distribution ellipsoid. At each step, this covariance matrix is used to generate a number of solution candidates. These candidates are evaluated for the function we are trying to optimize and then used to update the mean and covariance matrix of the multivariate Gaussian distribution. The step size for each update is also changed during the algorithm, by comparing the current path length - how much we change the mean of the distribution - with the expected one if steps were independent. If the path length is shorter, the steps are anti-correlated and the step size should be decreased, if the path length is longer it should be increased. To update the mean, a weighted sum of part (or all) of the new samples is done, where weights are higher for better samples and the chosen step size is taken into account. For the covariance, the new covariance matrix will depend both on a weighted sum of the new samples and on the evolution path, the sum of past steps for the mean.

NGOpt: The second black-box optimizer we used is NGOpt. An older version was presented in [Meu+21] (in which NGOpt8 is called ABBO), more recent modifications haven't been documented in peer-reviewed papers but were incorporated in the NGOpt optimizer available in the `Nevergrad` Python module. NGOpt is an algorithm selection technique, which will select the best algorithm to run on a problem. It selects the algorithm(s) to run based on features known in advance - dimension of the problem, type of variables, bounds for the variables or the evaluation budget for example - as well as based on information obtained during the execution of the algorithms. It can run multiple algorithms in parallel before only continuing with the best one, or run multiple algorithms sequentially. CMA-ES is one of the algorithms that NGOpt can select, but given our problem setting (dimension below 10, budget 1000), it does not call the same version of CMA as our CMA experiment. Given the lack of information on the function we are minimizing as well as our lack of budget for hyperparameter tuning and algorithm configuration, using NGOpt's intrinsic algorithm choice is a valuable help in guaranteeing that our chosen optimizers should be good.

In our problem formulation, potential solutions are of the shape $(p_i)_{i\in\{1,\dots,k\}}$ where, for all $i \in \{1, \dots, n-1\}$, $i < j$ implies $p_i \leq p_j$. CMA-ES and a large part of the algorithms in NGOpt try to learn some relation between the variables. For this, they use several evaluated points and update the sampling distribution. Most of these optimizers do not work well with constraints (they typically sample points

until they fit the constraints, potentially spending a lot of time), therefore our approach was to reorder the solution parameters to always have a sorted point set. After evaluating the fitness of a *sorted* candidate point, the optimizer would consider this to be the fitness of the *non-sorted* original point. The distribution update could then be misled by this sorting.

(1+1)-Evolutionary Strategy: To limit the impact of the above problem, we also run our experiments with a more traditional (1+1)-Evolutionary Strategy (ES). At each step, the current solution is modified by adding a random variable (in our case a Gaussian random variable) to each element of the solution, scaled by a varying step size. The best solution between the new one and the old solution is then kept for the next step. The step size is modified depending on the mutation success, according to the 1/5-th rule [SS68]: if too many steps are successful, the step size is increased, and it is decreased otherwise. This iterative process continues until we reach the given budget. There are many different versions of (1+1)-Evolutionary Strategies generally changing the mutation rates. We will be using here the `Parametrized(1+1)` implementation from the `Nevergrad` package with a Gaussian mutation.

10.4 Finding Minimal Sets

In this section, we use several different black-box optimizers to look for approximations of the minimal sets $\mathcal{S}^*(n,2)$ and $\mathcal{S}^*(n,3)$, which we then compare to our equivolume stratification. We will consider values $0 \leq p_1 \leq \cdots \leq p_{n-1} \leq \sqrt{d}$. These values correspond to the Euclidean distance between the origin and the intersection points of the chosen hyperplanes with the diagonal, and are more convenient to use with the optimizers than the hyperplane parameters appearing in the mathematical formulas. Black-box optimizers generate a number of samples, evaluate the target function for these samples (in this case, the expected L_2 discrepancy) and then update some information on the problem. This depends heavily on the type of optimizer, it can be only the best solution found so far or an underlying distribution which can be used to find the best solution.

10.4.1 Experiment Setup

Regardless of the chosen optimizer, the general model is identical. In dimension d, we have an $n-1$ dimensional problem, where each parameter is in the interval

Table 10.1: A percentage change comparison of the expected L_2 discrepancies of different point sets with respect to the equivolume stratification: uniformly distributed random points $R(n, 2)$, the equivolume partition corresponding to $\mathcal{S}(n, 2)$ and the best sets obtained by the three black-box optimizers. Expected L_2 discrepancy values are done with 10 000 repetitions to guarantee higher accuracy. We refer to Table 12.20 in Section 12.3 for the exact expected discrepancy values.

N	R(N, 2)	S(N, 2)	CMA-ES	NGOpt	(1+1)-ES
3	58.18%	-	-7.58%	-8.06%	4.94%
4	69.09%	-	-7.47%	-7.08%	13.02%
5	77.82%	-	-6.86%	-6.15%	-5.64%
6	82.94%	-	-6.37%	-6.05%	-6.29%
7	85.98%	-	-5.89%	-5.98%	0.93%
8	86.26%	-	-5.22%	-5.80%	-3.65%
9	90.10%	-	-4.92%	-4.60%	-3.80%
10	105.34%	-	-4.20%	-4.50%	-2.96%
15	96.38%	-	-1.64%	-1.64%	-2.09%
20	96.91%	-	0.25%	0.74%	-1.53%

$[0, \sqrt{d}]$. The optimizer is given a budget of 1000 evaluations[67] where each evaluation consists in approximating the expected L_2 discrepancy of the corresponding stratification. To approximate this function, we first sort the different parameters to obtain a valid stratification. We sample randomly a point in each stratum then compute the L_2 discrepancy of the resulting point set. This is repeated 1500 times and averaged to obtain an approximation of the expected L_2 discrepancy.

All experiments were run in dimension 2 and 3 in a low-fidelity setting, with a high-fidelity correction at the end. Indeed, since the optimizers are tracking only the best value found so far, it is possible that the returned point sets have slightly higher discrepancy and the calculation during the optimizer run was one with a high variance. To correct for this phenomenon, we recompute the expected L_2 discrepancies of the corresponding point sets, this time with 10 000 repetitions to guarantee higher accuracy. No initialization is provided to the optimizers and 10 runs are made for each instance.

Note that we could extend our experiments beyond dimension 3. The main difficulty is sampling uniformly from given hypercube slices. In our experiments, we define a slice based on the distance between two successive points p_i and p_{i+1}, sample a point in it before rotating and translating this slice into the right position

67 While this may seem low to practitioners, there appears to be very quick convergence. The solution is within noise range of the optimal after 100 evaluations.

to obtain the desired point. Note that re-sampling may be necessary as usually not all of the slice is indeed contained in the unit cube. However, this is in general not a problem. For $d = 3$ we use quaternions to define the rotations. In higher dimensions this gets more involved.

Both the `ioh` [Nob+24] and `Nevergrad` [RT18] Python packages were used for the optimizers, the implementations are taken from the `Nevergrad` package. Experiments were run in Python, using the `random` module to generate the points. The kernel density estimations were obtained using the `sklearn` package and in particular `Kernel Density`.

10.4.2 Experimental Results

Table 10.1 gives a percentage comparison of the expected L_2 discrepancy of the equivolume partition $\mathcal{S}(n, 2)$, of uniformly distributed random points $R(n, 2)$, and of the best distributions found by the different optimizers, using $\mathcal{S}(n, 2)$ as a baseline. The discrepancy estimation is obtained with 10 000 repetitions.

As proven in [Pau23], the expected L_2 discrepancy of the equivolume partition is (asymptotically) smaller by a factor of 2 than the discrepancy of uniformly distributed random points for all different set sizes tested. All three black-box optimizers return very similar values, which are about 7% smaller for the lowest values of n after re-evaluation of the expected L_2 discrepancy. Since all three optimizers return similar values, this suggests that the point ordering in the problem structure was not an issue, at least for these low values of n. The only outliers are for very low values of n, where the (1+1)-ES is struggling: this seems to be largely due to the low-fidelity optimization, the best discrepancy values found were around 0.025 for $n = 3$ before correction. We also note that for $n = 4$, both the discrepancy values and the point sets returned are quite similar to the improved construction given in [KP21] (see Table 10.2 for the point sets). The equivolume partition performs better than the optimizers for $n = 20$ and its relative performance improves when n increases in general.

As mentioned previously, discrepancy values for the point sets returned by the optimizers were recomputed with a higher precision. This implies that point sets with potentially smaller discrepancy may be *overlooked* during the optimization because of the imprecision in the expected L_2 discrepancy calculations. The gradually decreasing gap, both for the returned discrepancy value and the corrected one, also suggests that our optimizers may be struggling if the number of points is increased.

Table 10.2 gives examples of the obtained point sets (see also Section 12.3).

Interestingly, all the sets returned by the three optimizers have a similar structure

Table 10.2: Optimal partitioning points returned by CMA-ES. Note that we need $n-1$ hyperplanes for a set of n points.

n	Point set obtained by CMA-ES
3	[0.525326, 1.094506]
4	[0.416424, 0.880939, 0.995149]
5	[0.394111, 0.617818, 0.967749, 1.021636]
6	[0.346292, 0.547769, 0.833453, 0.916769, 1.066748]
7	[0.325707, 0.506785, 0.6853, 0.876577, 0.97997, 1.087066]
8	[0.322496, 0.482121, 0.583169, 0.863154, 0.874371, 0.981856, 1.066117]
9	[0.293892, 0.448608, 0.525631, 0.728415, 0.864673, 0.902607, 0.992526, 1.097269]
10	[0.295754, 0.447257, 0.49142, 0.603255, 0.805188, 0.88467, 0.923663, 1.005221, 1.120071]
15	[0.279248, 0.27951, 0.434256, 0.541966, 0.560045, 0.623614, 0.7019, 0.79564, 0.798366, 0.814506, 0.931907, 0.958269, 1.010601, 1.333425]
20	[0.14877, 0.239469, 0.432346, 0.471298, 0.487274, 0.50509, 0.545568, 0.58021, 0.612064, 0.737669, 0.802694, 0.856641, 0.889084, 0.921972, 0.946705, 0.977818, 1.009777, 1.022098, 1.259538]

once n is big enough ($n \geq 5$). This structure can be visualised using a *kernel density estimate* with a Gaussian kernel as illustrated in Figure 10.3.

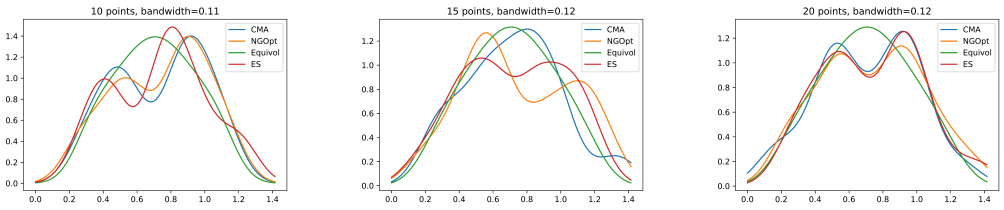

Figure 10.3: Kernel density estimate plots with Gaussian kernels for the best partitioning points obtained by the optimizers in dimension 2 as well as the equivolume sets for $n = 10, 15, 20$.

The point sets seem to have two clusters of points on the diagonal. The spread of the points seems bigger meaning that the first point has a smaller value than the first of the equivolume set, while the last point has a larger value than the corresponding point of the equivolume set.

Turning to dimension 3, Table 10.3 gives the percentage comparison of the expected L_2 discrepancy of the different methods with $\mathcal{S}(n, 3)$ as the baseline. The results are quite similar to the $d = 2$ case, where all three black-box optimizers return similar values, all outperforming the equivolume stratification and with the (1+1)-ES giving the worst results. Once again, we notice that the relative performance of the equivolume strategy improves as n increases. The plots in Figure 10.4 are

similar to corresponding plots in dimension 2: there are clusters of strata and the spread of the points seems larger. However, the last diagonal point p_{n-1} in the optimizers' sets is not always larger than the last diagonal point for $\mathcal{S}^*(n,3)$. We visualise the point sets again with a kernel density estimate in Figure 10.5.

Table 10.3: A percentage change comparison of the expected L_2–discrepancy values of different point sets using $\mathcal{S}(n,3)$ as a baseline. Discrepancy values are done with 10 000 repetitions; we refer to Table 12.21 in Section 12.3 for the exact expected discrepancy values.

N	R(N, 3)	S(N, 3)	CMA-ES	NGOpt	(1+1)-ES
3	34.09%	-	-10.91%	-10.55%	-10.05%
4	41.67%	-	-8.14%	-8.27%	-8.14%
5	45.45%	-	-8.51%	-8.76%	-8.02%
6	48.45%	-	-9.90%	-9.94%	-8.58%
7	49.52%	-	-8.68%	-7.68%	-7.66%
8	51.31%	-	-7.84%	-7.68%	-7.28%
9	54.53%	-	-6.67%	-7.05%	-6.77%
10	53.33%	-	-6.32%	-6.21%	-6.44%
15	57.98%	-	-5.08%	-5.29%	-4.81%
20	57.86%	-	-3.32%	-3.86%	-3.29%

Overall, while the equivolume stratification provides smaller expected L_2 discrepancy than random uniformly distributed points, our results suggest that it can still be improved. Finally, it is important to note that our black-box optimizers seem to return sets with similar structure as illustrated in Figure 10.3 and Figure 10.5 while the individual points in the respective sets are quite different; see the tables in Section 12.3. This suggests a plateau-like structure of the underlying space which is a blessing and a curse at the same time. On the one hand, we can safely assume that we found almost optimal solutions, on the other hand it leaves little hope to actually determine the minimal set.

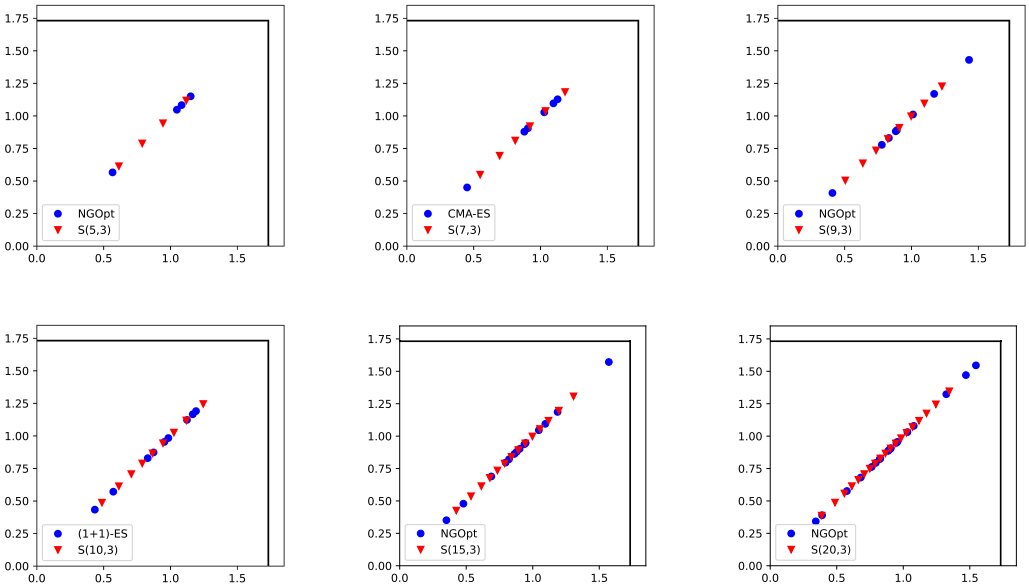

Figure 10.4: Best partitions generating stratified point sets obtained by the optimizers in dimension 3 compared to the equivolume partition for $n = 5, 7, 9, 10, 15, 20$.

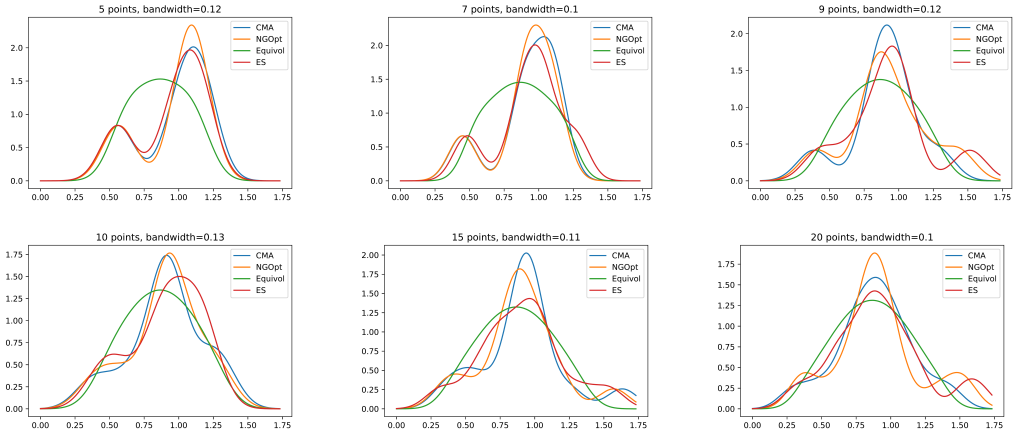

Figure 10.5: Kernel density estimate plots with Gaussian kernels for the best partitioning points obtained by the optimizers in dimension 3 as well as the equivolume sets for $n = 5, 7, 9, 10, 15, 20$.

Open questions This chapter presents results on optimal stratification of $[0,1]^2$ and $[0,1]^3$ for the class of partitions whose partitioning hyperplanes are orthogonal to the main diagonal of the unit cube. Hence,

1. how much can one improve upon the expected discrepancy values of the stratified point sets obtained in this text by relaxing some, or all of the constraints on the partitioning lines?

2. what is the optimal partition (with respect to the expected discrepancy of the resulting stratified point set) of $[0,1]^d$ for $d \geq 2$? Are the optimizers close to the mark, or are they missing a different class of solutions?

11 Future Work

In this thesis, we have proposed multiple computational approaches to tackle a number of discrepancy-related problems. Our key objective is dual: provide new empirical evidence to essential questions which have puzzled mathematicians for a long time, and introduce new optimization methods and tools to the discrepancy community. Our main focus has been on constructing new low-discrepancy point sets, in a wide variety of dimensions.

For smaller point sets in dimensions 2 and 3, we introduced non-linear programming formulations in Chapter 5 to provide exact constructions for the L_∞ star discrepancy and a set of other measures. We also showed that these methods could still be used to construct good points sets for up to 100 points. Subset selection increases the range for which we can obtain excellent sets, either via exact methods, Chapter 6, or via heuristics, Chapter 7. For exact methods, we can obtain good low-discrepancy sets in dimensions 2 and 3 up to 120 points. With the heuristic approaches, we are not limited by the dimension. We observe a 10 to 30% improvement on the initial set for all (n, d) combinations tried, though the results are poorer when n becomes too large compared to the number of points removed k. Finally, greedy approaches with the L_∞ or L_2 star discrepancy described in Chapter 8 enable the adaptation of a given set to a more sequence-oriented setting, allowing the addition of one or multiple points at a time to this set. This does remain applicable only to one, two or three dimensions for the moment.

Our methods should also not be seen as separate. While our work so far has kept them isolated, it is possible to generate a good set of points via the Kritzinger construction or from an unfinished optimization run, on which we can then use subset selection. We conclude this thesis with a short description of some open questions.

Optimal point sets: Our results are so far limited to around 20 points in dimension 2 and less than 10 in dimension 3. While we do not expect to push this much further, excellent (but not necessarily optimal) point sets can be obtained by simplifying the model. For example, our current work with the same co-authors as [Clé+23a] suggests that one can enforce the chosen permutation of points in model (5.6), i.e. fix the $a_{i,j}$ variables. In this way, we are able to obtain point sets of 100 points in dimension 2 with discrepancy 0.01525 with preliminary experi-

ments, against 0.027485 for the Fibonacci set and 0.01933 with our optimal model interrupted after 40 000 seconds (see Table 5.4).

A different direction is one mentioned to me by Nathan Kirk,[68] and that has also come up multiple times in questions in workshops and conference: using machine learning models for our numerous optimization problems. It appears that machine learning models based on a smoothed L_2 discrepancy[69] are able to give excellent point sets in a wide variety of dimensions and with point sets up to hundreds of points. Once again, the strong performance of a model based on the L_2 discrepancy reinforces our belief that the Kritzinger sequence is a low-discrepancy sequence (as the numerical evidence in Chapter 8 suggests for low dimensions.). It would not be surprising to see machine learning methods be successful in tackling a number of the problems introduced in this thesis. Albeit in a limited scope, our results with black-box optimizers in Chapter 10 already show that some approaches are able to learn structures relevant to discrepancy.

It will also be very interesting to see if the new point sets that can be generated with the methods described in this thesis actually translate to better empirical performances for the different applications, of which numerical integration is naturally the first. The sets obtained via subset selection did not perform noticeably better than the Sobol' sequence for numerical integration, and were less regular when projected to lower-dimensions. Do we also observe the same performance on applications with the sets obtained via optimization methods, or is there a qualitative difference compared to previous sets? Perhaps L_2-based approaches and their greater smoothness could provide more flexibility in the point sets than L_∞ approaches.

Multiple-corner discrepancy: Continuing on this last point, we introduced in Section 5.3.3.3 the multiple-corner discrepancy, to provide more symmetry to the optimal sets obtained. We showed that we could compute optimal sets for this measure at very little added cost relative to the original L_∞ star discrepancy setting. Should the new optimal constructions not perform as well as their excellent discrepancy might suggest, it could be very promising to research this measure further. It would also naturally be more adapted to any problem with symmetries, and be far less costly to compute (or design point sets for) than the extreme discrepancy. From a construction perspective, very simple experiments have allowed us to construct excellent L_∞ star discrepancy sets with a larger number of

68 This paragraph is based on personal communication with Nathan Kirk, whose work was not yet available online at time of writing

69 The smoothing comes from the minima appearing in the formula, as the model requires a derivable function.

points from the optimal multiple-corner discrepancy sets. Multiple copies of the optimal 16 point multiple-corner set placed together, with some rotations, allowed us to build sets of 64 points competitive or better than the Fibonacci set of similar size.

Subset selection: In a similar vein as for optimal sets, the main challenge for point sets obtained via subset selection is to make a difference in applications. Can one find an application where these sets are noticeably better than traditional sets and sequences (and, maybe soon, than the machine learning/optimized sets)? Some quick experiments on numerical integration did not show a clear improvement. In particular, as we focus on a single discrepancy measure when optimizing, the point sets tend to be less regular for other measures. For example, the discrepancy of the lower-dimensional projections of our sets is more often than not worse than that of the projections of the Sobol' sequence.[70]

This should be relatively easy to correct: one could explore a multi-objective approach in the subset selection heuristic,[71] or consider a weighted discrepancy function. The weighted discrepancy consists in a weighted sum of the discrepancies of all the different projections of the point set, introduced in [SW98]. For example, in dimension 3, there would be 3 terms corresponding to the one-dimensional discrepancy of the one-dimensional projections, 3 for the two-dimensional discrepancies of the two-dimensional projections and one for the discrepancy of the original set. Weights can be set as desired to simplify the formula (for example keeping only the two-dimensional terms and the original one) or to emphasize the importance of desired terms. As the L_∞ star discrepancy, the general weighted star discrepancy is still very expensive to compute. Nevertheless, there exist methods linked to lattice constructions that consider this measure when constructing the lattice, for example [LEc+22]. In our case, considering only a small specific set of projections such as all the two-dimensional ones could already be interesting.

The subset selection heuristic should not be seen as a rigid method, but as a general approach. In low and moderate dimensions, it is possible to run the heuristic with a tailored (weighted) discrepancy function for each experiment for a relatively low cost. This adaptation should also be possible in general when optimizing point sets. Multi-objective methods can be efficiently represented via constraints, or penalties in the objective function. Initial experiments suggest these methods work very well, with a lot of flexibility on the choice of these

70 It even seems that sets obtained with Steinerberger's functional are better in this regard.

71 This could take into account other objectives such as the worst 2-dimensional projection discrepancy value, the L_2 discrepancy or simply geometric characteristic of the points selected such as the dispersion or the average distance between points.

constraints.[72] Nevertheless, we expect subset selection to be more flexible with respect to other discrepancy measures and higher dimensions.

Algorithmic questions: From our work, there remain two complexity questions we were not able to solve: is L_∞ subset selection W[1]-hard, and is L_2 subset selection NP-hard? We expect the first to be true, and we admit not much time was put into this. Simply showing that at least one discrepancy calculation is necessary to solve subset selection would suffice to show it is W[1]-hard. We also expect the second to be true, as it seems that the linear dependency on the dimension would make this problem far too powerful in higher dimensions. The difficulty is not so much in finding NP-hard problems that can describe the problem, but in showing that the instances resulting from this representation are hard ones for the NP-hard problems.

On the discrepancy calculation side, we provided a parallelized version of the DEM algorithm. It seems unlikely to us that this algorithm could be improved much further, despite some recent minor improvements in [Cha13] to the complexity of solving Klee's Measure Problem, that had initially led to the development of the DEM algorithm. On the other hand, there seems to be plenty of improvement room on the heuristics side. As a continuation of Chapter 9, discussions with Olivier Teytaud suggest that multi-modal black-box optimizers perform extremely well, at least up to dimension 10. It remains to be seen exactly how far these can be pushed, and whether they can provide good solutions with a decent runtime. We also expect that the TA heuristic itself can be improved. Indeed, much progress has been made on multi-modal algorithms since its creation. This question would be of great practical and theoretical interest.

The Kritzinger sequence: We presented in Chapter 8 extended numerical results on the Kritzinger sequence. The natural question is what is its asymptotic discrepancy order. Having such a different construction method compared to other low-discrepancy sequences is fascinating, and obtaining a formal proof of the discrepancy order for this function or, even better, a class of functions, would be a great development in our understanding of uniformly distributed point sets. From a computational perspective, we would also not be surprised if better methods were available to compute the sequence. The $F(y, P)$ function seems to be very smooth in two dimensions when y varies, and simple local search algorithms could

72 One can for example find the optimal set such that the minimum distance between two points is maximal, or enforce that no two points are within a certain distance of each other.

perform well. We expect this smoothness to persist in higher dimensions.

To conclude, we point out that there has been growing interest in metrics designed to compare distributions, largely led by applications in machine learning. In particular, the field of optimal transport and the associated Wasserstein metrics has seen renewed interest, with a few hundred related preprints in the last few months. It would be of great interest to find the relations between the discrepancy measures studied in this thesis and these other measures. Indeed, the result on the Kritzinger sequence by Steinerberger in [Ste24] shows that the different approaches are not entirely dissimilar.

12 Appendix

This final chapter groups together a large set of data tables associated with the previous chapters.

12.1 Computational Results of Chapter 6

The tables on the following pages present the details of the observations summarized in the main body of Chapter 6.

Table 12.1: CPU-time (minimum, median, maximum and number of successful runs out of 10) taken by branch and bound, for several values of n and k for iLHS and unif point sets in the $3d$ case, where "-" indicates that the approach did not terminate before the time limit of 1800 seconds.

k	n	iLHS sequences min	med	max	(*succ*)	unif sequences min	med	max	(*succ*)
20	40	0	161	1742	(9)	1	31	464	(9)
	60	86	620	1305	(8)	140	305	569	(7)
	80	567	567	567	(1)	966	966	966	(1)
	100	-	-	-	(0)	-	-	-	(0)
40	60	253	872	1492	(2)	-	-	-	(0)
	80	-	-	-	(0)	-	-	-	(0)
	100	-	-	-	(0)	-	-	-	(0)
60	80	92	806	1519	(2)	25	86	147	(2)
	100	-	-	-	(0)	-	-	-	(0)
80	100	-	-	-	(0)	.	-	-	(0)

Table 12.2: Integrality gap of the LP relaxation of MILP for two-dimensional deterministic sequences and sets with respect to the optimal found and, when not available, with respect to the best solution found (marked with "*").

k	sequence	$n = 40$	$n = 60$	$n = 80$	$n = 100$	$n = 120$	$n = 140$
20	Faure	1.3556	1.6926	2.5039	2.4591	2.4024	4.3098
	Sobol	1.3556	2.2080	2.3290	2.7040	3.6873	4.6088
	Halton	1.2828	1.5105	2.8167	2.7912	2.7148	3.6817
	RevHal	1.6044	1.9179	2.7750	2.7750	2.7111	4.9240
	Fibon	2.1678	2.9110	3.6353	4.4520	5.1599	5.8107
40	Faure		1.1263	1.5811	1.5055	1.4535	*2.6538
	Sobol		1.4267	1.4695	1.7000	*2.2972	*2.7916
	Halton		1.0000	1.7417	1.7000	*1.6681	*2.2615
	RevHal		1.2107	1.7750	1.6889	1.6458	*3.0579
	Fibon		1.8513	2.2650	2.7402	3.1938	*3.5181
60	Faure			1.1971	1.1570	*1.1285	*2.0154
	Sobol			1.1304	1.2942	*1.6949	*2.1389
	Halton			1.3167	1.2833	*1.2750	*1.7231
	RevHal			1.3439	1.3063	*1.4222	*2.2702
	Fibon			1.6114	2.0355	*2.4153	*2.7430
80	Faure				1.0543	*1.0016	1.6392
	Sobol				1.1004	*1.4218	*1.5409
	Halton				1.0718	*1.0194	*1.4077
	RevHal				1.1250	*1.0222	*1.8906
	Fibon				1.6066	*1.9283	*2.2117
100	Faure					1.0000	1.4063
	Sobol					1.2469	*1.2034
	Halton					1.0000	*1.2434
	RevHal					1.0000	*1.5956
	Fibon					1.6097	*1.8532
120	Faure						1.2055
	Sobol						1.0000
	Halton						1.1077
	RevHal						1.3565
	Fibon						1.5534

Table 12.3: CPU-time in seconds obtained by the ILP solver on the MILP formulation and by BB, for several values of n and k for $2d$ low-discrepancy sequences and sets, where "-" indicates that the approach did not terminate before the time limit.

k	sequence	$n = 40$		$n = 60$		$n = 80$		$n = 100$		$n = 120$		$n = 140$	
		MILP	BB	MILP	BB	MILP	BB	MILP	BB	MILP	BB	MILP	BB
20	Faure	4	0	25	0	82	2	967	11	-	19	-	83
	Sobol	5	0	19	0	114	2	740	5	-	15	-	60
	Halton	6	0	26	0	89	2	-	14	-	44	-	72
	RevHal	4	0	28	0	87	1	-	10	-	32	-	102
	Fibon	4	0	45	0	-	2	-	26	-	20	-	67
40	Faure			43	1	169	24	-	215	-	1795	-	-
	Sobol			19	0	479	45	-	374	-	-	-	-
	Halton			9	8	294	13	-	216	-	-	-	-
	RevHal			10	1	214	15	1560	216	-	1235	-	-
	Fibon			53	0	284	18	-	187	-	-	-	-
60	Faure					33	14	-	496	-	-	-	-
	Sobol					41	56	-	200	-	-	-	-
	Halton					135	8	-	1106	-	-	-	-
	RevHal					47	10	-	761	-	-	-	-
	Fibon					143	7	-	518	-	-	-	-
80	Faure							161	1194	-	-	-	-
	Sobol							254	123	-	-	-	-
	Halton							843	161	-	-	-	-
	RevHal							517	305	-	-	-	-
	Fibon							1608	45	-	-	-	-
100	Faure									19	-	-	-
	Sobol									1538	847	-	-
	Halton									12	-	-	-
	RevHal									12	-	-	-
	Fibon									-	104	-	-
120	Faure											321	-
	Sobol											127	-
	Halton											1332	-
	RevHal											491	1253
	Fibon											-	915

Table 12.4: CPU-time (minimum, median, maximum and number of successful runs out of 10) of the ILP solver on the MILP formulation, and BB, for several values of n and k for iLHS and unif point sets in the $2d$ case, where "-" indicates that the approach did not terminate before the time limit of 1800 seconds.

		iLHS sequences								unif sequences							
k	n	MILP				BB				MILP				BB			
		min	med	max	(*succ*)	min	med	max	(*succ*)	min	med	max	(*succ*)	min	med	max	(*succ*)
20	40	2	2	4	(10)	0	0	0	(10)	0	1	1	(10)	0	0	11	(10)
	60	18	30	45	(10)	0	0	0	(10)	1	16	41	(10)	0	2	201	(10)
	80	71	140	190	(9)	1	2	5	(10)	15	86	458	(10)	2	12	134	(10)
	100	689	913	1323	(7)	8	13	231	(10)	281	1122	1375	(6)	4	21	230	(10)
	120	1486	1486	1486	(1)	15	30	49	(10)	1343	1555	1768	(2)	21	91	252	(10)
	140	-	-	-	(0)	75	107	185	(10)	-	-	-	(0)	35	210	348	(10)
40	60	5	17	26	(10)	0	0	3	(10)	0	2	4	(10)	2	192	922	(6)
	80	101	160	247	(10)	5	12	19	(10)	3	6	20	(10)	81	94	107	(2)
	100	-	-	-	(0)	64	143	265	(10)	11	111	1045	(9)	3	615	1227	(2)
	120	-	-	-	(0)	413	1089	1488	(7)	7	376	1424	(8)	185	185	185	(1)
	140	-	-	-	(0)	-	-	-	(0)	8	8	8	(1)	678	678	678	(1)
60	80	15	31	62	(10)	3	7	104	(10)	2	3	51	(10)	0	287	1691	(3)
	100	355	501	1707	(10)	110	195	643	(10)	3	9	66	(10)	-	-	-	(0)
	120	-	-	-	(0)	1030	1220	1411	(2)	4	95	500	(10)	-	-	-	(0)
	140	-	-	-	(0)	-	-	-	(0)	76	227	1349	(8)	-	-	-	(0)
80	100	75	127	273	(10)	22	59	652	(10)	3	4	7	(10)	0	3	4	(3)
	120	292	881	1459	(6)	759	1423	1774	(4)	4	14	233	(10)	-	-	-	(0)
	140	-	-	-	(0)	-	-	-	(0)	7	28	191	(9)	-	-	-	(0)
100	120	23	167	324	(10)	24	120	365	(6)	3	5	18	(10)	5	5	5	(1)
	140	565	1017	1703	(3)	-	-	-	(0)	6	10	18	(10)	27	27	27	(1)
120	140	35	259	528	(10)	796	853	1114	(4)	5	8	19	(10)	-	-	-	(0)

Table 12.5: 2*d*, low-discrepancy sequences and sets: best found star discrepancy values found by *random* subset sampling, by the *greedy* heuristic, compared to the optimal or the best found (marked with *) values returned by MILP or BB (column *subset*), for all tested combinations of n and k. Best star discrepancy values for each (n,k) combination are underlined, and the minimum for each k is highlighted in **boldface**.

k	seq.	$n = k$	$n = 40$			$n = 60$			$n = 80$			$n = 100$			$n = 120$			$n = 140$		
			rand.	*greed.*	*subset*	*rand.*	*greed.*	*subset*	*rand.*	*greed.*	*subset*	*rand.*	*greed.*	*subset*	*rand.*	*greed.*	*subset*	*rand.*	*greed.*	*subset*
20	Faure	0.2094	0.0911	0.1169	0.0834	0.0994	0.1656	0.0785	0.0922	0.1305	0.0776	0.1036	0.1305	0.0762	0.1018	0.1554	0.0745	0.1038	0.1554	0.0738
	Sobol'	0.1313	0.0938	0.1254	0.0834	0.0960	0.1656	0.0809	0.1000	0.1472	0.0785	0.1014	0.1472	0.0743	0.1029	0.1554	0.0743	0.1031	0.1905	0.0738
	Halton	0.1477	0.0944	0.1681	0.0861	0.1000	0.1463	0.0833	0.0979	0.1537	0.0782	0.0965	0.1537	0.0775	0.1025	0.1315	0.0754	0.1029	0.1315	0.0739
	RevHal	0.1500	0.0935	0.1375	0.0836	0.0977	0.1639	0.0829	0.1000	0.1241	0.0771	0.1012	0.1421	0.0771	0.1059	0.1335	0.0753	0.1039	0.1481	0.0736
	Fibon	0.0930	0.0931	0.1257	0.0866	0.0971	0.1390	0.0828	0.1000	0.1598	0.0790	0.1023	0.1779	0.0757	0.1009	0.1506	0.0741	0.1038	0.1449	**0.0731**
40	Faure	0.0836				0.0590	0.0747	0.0523	0.0656	0.0832	0.0490	0.0691	0.0945	0.0467	0.0722	0.1154	0.0451	0.0714	0.1154	*0.0454
	Sobol'	0.0836				0.0613	0.0695	0.0522	0.0666	0.0899	0.0495	0.0656	0.0899	0.0467	0.0697	0.1154	*0.0463	0.0687	0.1117	***0.0447**
	Halton	0.0993				0.0611	0.1162	0.0552	0.0656	0.0972	0.0484	0.0696	0.1157	0.0472	0.0729	0.1157	*0.0463	0.0719	0.1157	*0.0454
	RevHal	0.0866				0.0628	0.0880	0.0523	0.0667	0.0841	0.0493	0.0699	0.0841	0.0469	0.0703	0.1021	0.0457	0.0730	0.0985	*0.0457
	Fibon	0.0545				0.0583	0.0726	0.0498	0.0656	0.1008	0.0485	0.0669	0.0807	0.0475	0.0713	0.0862	0.0463	0.0714	0.0742	*0.0449
60	Faure	0.0645							0.0464	0.0705	0.0371	0.0522	0.0726	0.0359	0.0540	0.0872	*0.0350	0.0564	0.0937	*0.0345
	Sobol'	0.0484							0.0472	0.0659	0.0381	0.0510	0.0807	0.0356	0.0540	0.0820	*0.0341	0.0562	0.0917	*0.0343
	Halton	0.0654							0.0453	0.0644	0.0366	0.0516	0.0625	0.0357	0.0539	0.0604	*0.0354	0.0561	0.0609	*0.0346
	RevHal	0.0626							0.0468	0.0646	0.0391	0.0500	0.0583	0.0363	0.0530	0.0667	*0.0354	0.0561	0.0661	***0.0339**
	Fibon	0.0363							0.0436	0.0826	0.0364	0.0498	0.0925	0.0345	0.0531	0.0866	*0.0345	0.0537	0.0720	*0.0344
80	Faure	0.0452										0.0397	0.0472	0.0327	0.0432	0.0468	*0.0311	0.0448	0.0433	0.0281
	Sobol'	0.0506										0.0398	0.0432	0.0302	0.0424	0.0439	*0.0286	0.0461	0.0546	*0.0285
	Halton	0.0375										0.0387	0.0500	0.0298	0.0424	0.0454	*0.0283	0.0435	0.0454	*0.0282
	RevHal	0.0454										0.0387	0.0426	0.0313	0.0430	0.0556	*0.0284	0.0439	0.0521	*0.0282
	Fibon	**0.0272**										0.0349	0.0674	0.0282	0.0395	0.0683	*0.0280	0.0432	0.0551	*0.0279
100	Faure	0.0461													0.0340	0.0471	0.0310	0.0348	0.0386	0.0241
	Sobol'	0.0398													0.0323	0.0471	0.0262	0.0365	0.0471	*0.0253
	Halton	0.0502													0.0329	0.0432	0.0299	0.0364	0.0532	*0.0250
	RevHal	0.0416													0.0330	0.0488	0.0299	0.0334	0.0566	*0.0238
	Fibon	0.0232													0.0298	0.0463	**0.0230**	0.0349	0.0531	*0.0232
120	Faure	0.0372																0.0273	0.0332	0.0211
	Sobol'	0.0251																0.0277	0.0329	0.0227
	Halton	0.0423																0.0292	0.0323	0.0222
	RevHal	0.0417																0.0279	0.0298	0.0213
	Fibon	0.0210																0.0254	0.0379	**0.0199**

Table 12.6: $2d$, iLHS: Minimum, median, and maximum of the best star discrepancy values found for ten independently generated iLHS point sets per each combination of k and n in $d = 2$. We show values returned by *random*, by *greedy*, and by the exact strategies (column *subset*). Where none of MILP or BB converged within the given time frame of 1800 seconds, the best found upper bound is shown (marked by a *); the number in parenthesis counts the point sets for which the optimal value could be computed by at least one of the two exact solvers. The best value found for a given instance is the minimum obtained by all exact approaches. The minimum value for each m is in **boldface**.

k	$k = n$			n	*random*			*greedy*			*subset*			
	min	med	max		min	med	max	min	med	max	min	med	max	(*succ*)
20	0.1022	0.1403	0.1714	40	0.0892	0.0936	0.0956	0.1201	0.1406	0.1647	0.0838 (M,B)	0.0866	0.0894	(10)
				60	0.0953	0.0969	0.0992	0.1186	0.1364	0.1580	0.0786 (M,B)	0.0817	0.0825	(10)
				80	0.0942	0.0993	0.1009	0.1226	0.1465	0.2022	0.0774 (M,B)	0.0785	0.0800	(10)
				100	0.0970	0.1022	0.1035	0.1256	0.1493	0.1897	0.0756 (B)	0.0770	0.0778	(10)
				120	0.0993	0.1042	0.1071	0.1146	0.1471	0.1952	0.0734 (B)	0.0748	0.0762	(10)
				140	0.1001	0.1034	0.1070	0.1293	0.1394	0.1876	**0.0731** (B)	0.0742	0.0747	(10)
40	0.0682	0.0779	0.1004	60	0.0588	0.0621	0.0640	0.0763	0.0889	0.0994	0.0507 (M,B)	0.0520	0.0556	(10)
				80	0.0601	0.0658	0.0676	0.0693	0.0900	0.1284	0.0478 (M,B)	0.0486	0.0496	(10)
				100	0.0663	0.0683	0.0711	0.0793	0.0975	0.1264	0.0465 (M,B)	0.0470	0.0477	(10)
				120	0.0696	0.0708	0.0724	0.0743	0.0905	0.1209	0.0452 (B)	0.0460	0.0465	(7)
				140	0.0719	0.0725	0.0743	0.0864	0.1041	0.1323	*__0.0445__ (B)	0.0450	0.0539	(1)
60	0.0508	0.0619	0.0680	80	0.0450	0.0466	0.0481	0.0562	0.0665	0.0882	0.0380 (M,B)	0.0396	0.0427	(10)
				100	0.0498	0.0504	0.0527	0.0606	0.0736	0.0919	0.0356 (M,B)	0.0363	0.0368	(10)
				120	0.0529	0.0541	0.0562	0.0555	0.0795	0.0974	*0.0343 (M,B)	0.0348	0.0354	(2)
				140	0.0543	0.0565	0.0581	0.0673	0.0760	0.1053	*__0.0338__ (B)	0.0345	0.0350	(0)
80	0.0389	0.0447	0.0567	100	0.0376	0.0386	0.0391	0.0460	0.0530	0.0616	0.0304 (M,B)	0.0317	0.0340	(10)
				120	0.0408	0.0425	0.0434	0.0474	0.0578	0.0876	0.0288 (M)	0.0291	0.0297	(7)
				140	0.0430	0.0449	0.0467	0.0512	0.0615	0.0715	*__0.0280__ (B)	0.0287	0.0300	(0)
100	0.0301	0.0388	0.04833	120	0.0322	0.0335	0.0341	0.0380	0.0433	0.0479	0.0259 (M,B)	0.0270	0.0290	(10)
				140	0.0355	0.0366	0.0378	0.0395	0.0460	0.0604	**0.0246** (M)	0.0252	0.0257	(3)
120	0.0280	0.0368	0.0436	140	0.0279	0.0294	0.0304	0.0314	0.0383	0.0462	**0.0229** (M,B)	0.0240	0.0270	(10)

Table 12.7: $2d$, unif: Minimum, median, and maximum of the best star discrepancy values found for ten independently sampled unif point sets per each combination of k and n in $d = 2$. We show values returned by *random*, by *greedy*, and by the exact strategies (column *subset*). Where none of MILP or BB converged within the given time frame of 1800 seconds, the best found upper bound is shown (marked by a *); the number in parenthesis counts the point sets for which the optimal value could be computed by at least one of the two exact solvers. The best value found for a given instance is the minimum obtained by all exact approaches for any of the ten point sets. The minimum value for each k is in **boldface**.

k	$k = n$			n	*random*			*greedy*			*subset*			
	min	med	max		min	med	max	min	med	max	min	med	max	(*succ*)
20	0.1836	0.2773	0.3450	40	0.1014	0.1143	0.1272	0.1403	0.1621	0.1943	0.0992 (*M*,*B*)	0.1139	0.1272	(10)
				60	0.1026	0.1081	0.1398	0.1288	0.1546	0.2334	0.0865 (*M*,*B*)	0.0956	0.1398	(10)
				80	0.1001	0.1053	0.1141	0.1405	0.1542	0.2080	0.0821 (*M*,*B*)	0.0854	0.0925	(10)
				100	0.0999	0.1049	0.1086	0.1232	0.1540	0.1947	0.0787 (*M*,*B*)	0.0803	0.0833	(10)
				120	0.1020	0.1055	0.1114	0.1366	0.1519	0.2259	0.0765 (*B*)	0.0778	0.0798	(10)
				140	0.1014	0.1083	0.1107	0.1218	0.1472	0.1777	**0.0753** (*B*)	0.0763	0.0775	(10)
40	0.1369	0.1648	0.2625	60	0.0771	0.0856	0.1077	0.0973	0.1293	0.1685	0.0678 (*M*,*B*)	0.0830	0.1077	(10)
				80	0.0757	0.0801	0.0938	0.0931	0.1094	0.1534	0.0624 (*M*,*B*)	0.0704	0.0938	(10)
				100	0.0757	0.0805	0.0846	0.0923	0.1133	0.1765	0.0531 (*M*,*B*)	0.0586	0.0643	(9)
				120	0.0753	0.0791	0.0835	0.0753	0.1048	0.1431	0.0498 (*M*)	0.0552	0.0737	(8)
				140	**0.0749**	0.0792	0.0846	0.0836	0.1105	0.1221	*__0.0479__ (*B*)	0.0494	0.0666	(2)
60	0.1205	0.1583	0.2123	80	0.0678	0.0750	0.1016	0.0838	0.1011	0.1231	0.0629 (*M*,*B*)	0.0745	0.1016	(10)
				100	0.0661	0.0712	0.1047	0.0790	0.1009	0.1474	0.0496 (*M*)	0.0580	0.1047	(10)
				120	0.0637	0.0683	0.0882	0.0754	0.0901	0.1114	0.0422 (*M*)	0.0527	0.0882	(10)
				140	0.0623	0.0687	0.0713	0.0849	0.0993	0.1244	**0.0404** (*M*)	0.0442	0.0498	(8)
80	0.0913	0.1343	0.2101	100	0.0569	0.0651	0.0853	0.0628	0.0825	0.1047	0.0569 (*M*)	0.0632	0.0853	(10)
				120	0.0565	0.0617	0.0836	0.0696	0.0817	0.1236	0.0427 (*M*)	0.0502	0.0787	(10)
				140	0.0565	0.0624	0.0645	0.0645	0.0789	0.1124	**0.0399** (*M*)	0.0450	0.0577	(9)
100	0.0946	0.1172	0.1774	120	0.0547	0.0706	0.1030	0.0732	0.0954	0.1316	**0.0452** (*M*)	0.0706	0.1030	(10)
				140	0.0546	0.0595	0.0748	0.0701	0.0756	0.1081	0.0481 (*M*)	0.0509	0.0748	(10)
120	0.0662	0.1116	0.1511	140	**0.0557**	0.0604	0.0692	0.0580	0.0721	0.0890	**0.0557** (*M*)	0.0588	0.0692	(10)

Table 12.8: 3*d*, low-discrepancy sequences: best found star discrepancy values found by *random* subset sampling, by the *greedy* heuristic, compared to the optimal or the best found (marked with *) values returned by MILP or BB (column *subset*), for all tested combinations of n and k. Best star discrepancy values for each (n,k) combination are underlined, and the minimum for each k is highlighted in **boldface**.

k	sequence	$n = k$	$n = 40$			$n = 60$			$n = 80$			$n = 100$		
			random	*greedy*	*subset*	*random*	*greedy*	*subset*	*random*	*greedy*	*subset*	*random*	*greedy*	*subset*
20	Faure	0.1795	0.1559	0.2206	0.1316	0.1612	0.2518	0.1205	0.1664	0.3428	*0.1223	0.1714	0.3193	*0.1225
	Sobol'	0.1774	0.1493	0.1758	0.1268	0.1616	0.1817	0.1220	0.1590	0.2028	***0.1202**	0.1692	0.2028	*0.1263
	Halton	0.2079	0.1635	0.1800	0.1311	0.1664	0.1917	*0.1240	0.1663	0.1912	*0.1214	0.1648	0.1990	*0.1244
	RevHal	0.1870	0.1511	0.1767	0.1300	0.1509	0.1956	0.1250	0.1635	0.1633	0.1207	0.1678	0.1801	*0.1242
40	Faure	0.1836				0.1007	0.1239	*0.0805	0.1024	0.1654	***0.0778**	0.1073	0.1781	*0.0801
	Sobol'	0.1066				0.1039	0.1172	*0.0817	0.1114	0.1323	*0.0810	0.1144	0.1311	*0.0786
	Halton	0.1475				0.1028	0.1425	0.0854	0.1015	0.1464	*0.0809	0.1083	0.1470	*0.0807
	RevHal	0.1333				0.1091	0.1441	0.0817	0.1091	0.1441	*0.0799	0.1119	0.14371	*0.0812
60	Faure	0.1107							0.0744	0.0900	***0.0606**	0.0857	0.0874	*0.0666
	Sobol'	0.0736							0.0834	0.0892	*0.0674	0.0875	0.0948	*0.0643
	Halton	0.1081							0.0775	0.0917	*0.0642	0.0842	0.0878	*0.0648
	RevHal	0.0866							0.0778	0.0883	*0.0654	0.0839	0.0882	*0.0648
80	Faure	0.0640										0.0661	0.0846	*0.0638
	Sobol'	0.0828										0.0655	0.0637	*0.0605
	Halton	0.0700										0.0640	0.0618	*0.0550
	RevHal	0.0747										0.0644	0.0792	***0.0547**

Table 12.9: $3d$, iLHS: Minimum, median, and maximum of the best star discrepancy values found for ten independently sampled iLHS point sets per each combination of k and n in $d = 3$. We show values returned by *random*, by *greedy*, and by the exact strategies (column *subset*). Where none of MILP or BB converged within the given time frame of 1800 seconds, the best found upper bound is shown (marked by a *); the number in parenthesis counts the point sets for which the optimal value could be computed by at least one of the two exact solvers. The best value found for a given instance is the minimum obtained by all exact approaches for any of the ten point sets. The minimum value for each k is in **boldface**.

k	$k = n$			n	*random*			*greedy*			*subset*			
	min	med	max		min	med	max	min	med	max	min	med	max	(*succ*)
20	0.3946	0.4325	0.4721	40	0.2008	0.2158	0.2759	0.1936	0.2261	0.2459	0.1462	0.1678	0.2184	(9)
				60	0.1911	0.2184	0.2421	0.1917	0.2132	0.2654	0.1388	0.1503	0.1933	(7)
				80	0.1958	0.2081	0.2333	0.1665	0.1918	0.2455	***0.1290**	0.1368	0.1553	(1)
				100	0.1974	0.2093	0.2221	0.1754	0.1984	0.2143	*0.1296	0.1383	0.1976	(0)
40	0.3003	0.3364	0.4545	60	0.1990	0.2441	0.2739	0.1618	0.1993	0.2303	*0.1515	0.1752	0.2302	(2)
				80	0.1896	0.2139	0.2395	0.1348	0.1543	0.1675	*0.1026	0.1211	0.1577	(0)
				100	0.1955	0.2013	0.2180	0.1283	0.1427	0.1667	***0.0977**	0.1333	0.1600	(0)
60	0.3352	0.3673	0.4173	80	0.2314	0.2583	0.2747	0.1639	0.1943	0.2336	0.1639	0.1906	0.2336	(2)
				100	0.1945	0.2211	0.2537	0.1061	0.1192	0.1547	***0.1056**	0.1157	0.1547	(0)
80	0.2851	0.3630	0.3792	100	0.2087	0.2640	0.2795	**0.1709**	0.2143	0.2305	***0.1709**	0.2143	0.2305	(0)

Table 12.10: 3d, unif: Minimum, median, and maximum of the best star discrepancy values found for ten independently sampled unif point sets per each combination of k and n in $d = 2$. We show values returned by *random*, by *greedy*, and by the exact strategies (column *subset*). Where none of MILP or BB converged within the given time frame of 1800 seconds, the best found upper bound is shown (marked by a *); the number in parenthesis counts the point sets for which the optimal value could be computed by at least one of the two exact solvers. The best value found for a given instance is the minimum obtained by all exact approaches for any of the ten point sets. The minimum value for each k is in **boldface**.

k	$k = n$			n	*random*			*greedy*			*subset*			
	min	med	max		min	med	max	min	med	max	min	med	max	(*succ*)
20	0.4089	0.4923	0.5606	40	0.2063	0.2423	0.2802	0.1952	0.2272	0.2603	0.1693	0.2017	0.2603	(9)
				60	0.1955	0.2109	0.2442	0.1657	0.2112	0.2487	0.1363	0.1452	0.1598	(7)
				80	0.1977	0.2126	0.2219	0.1686	0.2101	0.2975	*0.1338	0.1406	0.1813	(1)
				100	0.1999	0.2091	0.2272	0.1755	0.1966	0.2398	***0.1292**	0.1415	0.1470	(0)
40	0.3375	0.3889	0.4524	60	0.2100	0.2659	0.2921	0.1850	0.2358	0.2578	*0.1499	0.2358	0.2578	(0)
				80	0.2136	0.2219	0.2645	0.1279	0.1558	0.2601	*0.1168	0.1369	0.1748	(0)
				100	0.1898	0.2209	0.2504	0.1419	0.1712	0.2150	***0.1127**	0.1404	0.1691	(0)
60	0.3378	0.3992	0.4491	80	0.2354	0.2533	0.3181	0.1855	0.2367	0.3181	*0.1855	0.2367	0.3181	(0)
				100	0.2324	0.2599	0.2953	0.1234	0.1660	0.2625	***0.1187**	0.1614	0.2161	(0)
80	0.3190	0.3510	0.4197	100	0.2279	0.2795	0.3176	**0.1872**	0.2410	0.2801	***0.1872**	0.2410	0.2801	(0)

12.2 Computational Results for Chapter 7

We include in this section some of the discrepancy values obtained with the different methods in Chapter 7. Unless they are in **bold**, values from TA sets were verified with the DEM algorithm. This could lead to higher discrepancy values than those obtained during the subset selection heuristic, and possibly that this heuristic missed better sets because of those mistakes. The corrected values for the TA heuristics only include those from *concluded* runs. A large number of the bold values simply correspond to sets where the best run was interrupted and the stored set was clearly poorer.

Table 12.11: Discrepancy values obtained in dimension 4 for the different heuristics, the DEM_NBF version was not run for $n \geq 200$.

Set size	Subset size k	DEM_BF	DEM_NBF	TA_BF	TA_NBF
$n = 50$	$k = 50$	0.13422	0.13422	0.13422	0.13422
	40	0.12236	0.12520	**0.13406**	0.14090
	30	0.14020	0.14924	**0.15431**	0.15471
	20	0.17660	0.18494	**0.17883**	0.20066
$n = 100$	$k = 100$	0.092688	0.092688	0.092688	0.092688
	90	0.070093	0.075315	**0.076933**	0.075315
	80	0.071985	0.082731	**0.084182**	0.08625
	70	0.078701	0.087342	0.09545	0.088841
	60	0.087650	0.095528	**0.103849**	0.100801
	50	0.097189	0.110063	**0.119433**	0.118624
$n = 150$	$k = 150$	0.061738	0.061738	0.061738	0.061738
	140	0.052081	0.054116	0.056260	**0.054268**
	130	0.051702	0.057176	0.060173	0.060173
	120	0.056405	0.062694	0.065592	0.065592
	110	0.059261	0.063807	**0.067161**	0.070707
	100	0.061478	0.068499	**0.077275**	**0.077275**
$n = 200$	$k = 200$	-	0.050215	0.050215	0.050215
	190	-	0.045960	0.054374	0.045960
	180	-	0.046837	-	0.046848
	170	-	0.048267	0.054912	0.052956
	160	-	0.051588	0.065464	0.054244
	150	-	0.053195	-	**0.055839**
$n = 250$	$k = 250$	-	0.038216	0.038216	0.038216
	240	-	0.036286	0.040657	0.037994
	230	-	0.037675	0.040731	0.040603
	220	-	0.037972	-	**0.043796**
	210	-	0.039900	-	0.042615
	200	-	0.043015	-	0.046989
$n = 500$	$k = 500$	-	0.022901	-	0.022901
	490	-	0.021662	-	**0.021187**
	480	-	0.021572	-	**0.022823**
	470	-	0.021211	-	0.024016
	460	-	0.022744	-	0.025268
	450	-	0.023916	-	0.027314

Table 12.12: Discrepancy values obtained in dimension 5 for the different heuristics, the DEM_NBF version was not run for $n \geq 200$. The - in the DEM column indicate that not a single run finished.

Set size	Subset size k	DEM_BF	DEM_NBF	TA_BF	TA_NBF
$n = 50$	$k = 50$	0.165488	0.165488	0.165488	0.165488
	40	0.138741	0.149003	**0.148836**	0.147256
	30	0.161715	0.171163	0.17149	0.17149
	20	0.211900	0.214233	0.234375	0.23438
$n = 100$	$k = 100$	0.120707	0.120707	0.120707	0.120707
	90	0.086374	0.092956	**0.090522**	0.091191
	80	0.090923	0.095958	**0.099937**	**0.097624**
	70	0.099563	0.105703	**0.109832**	0.116787
	60	0.107044	0.117161	0.125410	**0.118965**
	50	0.118428	0.129599	**0.135023**	**0.134716**
$n = 150$	$k = 150$	0.074899	0.074899	0.074899	0.074899
	140	0.064423	0.054116	**0.054163**	**0.054268**
	130	0.064549	0.057176	**0.060173**	**0.060173**
	120	0.068790	0.062694	**0.063046**	**0.063046**
	110	0.073173	0.063807	**0.067161**	**0.067161**
	100	0.077755	0.068499	**0.077275**	**0.077275**
$n = 200$	$k = 200$	-	0.058292	0.058292	0.058292
	190	-	0.053908	0.059209	**0.053496**
	180	-	0.055425	0.061967	**0.058496**
	170	-	0.059826	0.067392	0.062506
	160	-	0.061195	0.075936	0.065796
	150	-	0.064438	-	**0.067725**
$n = 250$	$k = 250$	-	0.053507	0.053507	0.053507
	240	-	0.044187	0.048002	**0.048310**
	230	-	0.046281	0.046463	0.046463
	220	-	0.048236	0.053760	0.052865
	210	-	0.049783	-	**0.055719**
	200	-	0.052454	0.062576	**0.055284**
$n = 500$	$k = 500$	-	0.029117	-	0.029117
	490	-	0.028160	-	0.029485
	480	-	0.029255	-	0.030663
	470	-	0.030951	-	0.031702
	460	-	-	-	0.034878
	450	-	-	-	0.036084

Table 12.13: Discrepancy values obtained in dimension 6 for the different heuristics, the DEM_NBF version was not run for $n \geq 200$. Other - correspond to unfinished runs.

Set size	Subset size k	DEM_BF	DEM_NBF	TA_BF	TA_NBF
$n = 50$	$k = 50$	0.225548	0.225548	0.225548	0.225548
	40	0.162600	0.166336	**0.166522**	0.169385
	30	0.186518	0.197138	**0.197326**	0.194999
	20	0.232082	0.246097	0.26041	0.253279
$n = 100$	$k = 100$	0.124451	0.124451	0.124451	0.124451
	90	0.100532	0.102214	**0.100660**	0.11364
	80	0.109108	0.114143	**0.113169**	**0.114344**
	70	0.120823	0.120817	0.133813	0.120032
	60	0.128823	0.134782	**0.140478**	**0.131683**
	50	0.139858	0.149740	**0.161732**	**0.150491**
$n = 150$	$k = 150$	0.090827	0.090827	0.090827	0.090827
	140	-	0.081020	**0.078646**	**0.082801**
	130	-	0.085759	**0.078646**	**0.084272**
	120	-	0.089289	0.091492	**0.090094**
	110	-	0.090785	0.103782	**0.097381**
	100	-	0.100891	0.104589	0.107580
$n = 200$	$k = 200$	-	0.087784	-	0.087784
	190	-	0.068581	0.078203	**0.065424**
	180	-	0.070267	0.077221	**0.065122**
	170	-	-	0.081820	**0.071578**
	160	-	0.077993	-	**0.072786**
	150	-	0.084923	-	**0.079571**
$n = 250$	$k = 250$	-	0.088941	-	0.088941
	240	-	0.064764	-	**0.057167**
	230	-	-	-	**0.060105**
	220	-	-	-	**0.064285**
	210	-	-	-	**0.062227**
	200	-	0.073417	-	**0.068424**
$n = 500$	$k = 500$	-	0.040529	-	0.040529
	490	-	-	-	0.036165
	480	-	-	-	**0.034287**
	470	-	-	-	**0.038244**
	460	-	-	-	0.042268
	450	-	-	-	0.035382

Table 12.14: Discrepancy values obtained in dimension 8, 10, 15 and 25 for TA. DEM_NBF could finish only for the smallest k in dimension 8 and is omitted here. **None of the values were verified with the exact algorithm**.

Set size	Subset size k	TA_NBF $d = 8$	TA_NBF $d = 10$	TA_NBF $d = 15$	TA_NBF $d = 25$
$n = 50$	$k = 50$	0.248547	0.298001	0.388598	0.483923
	40	0.207029	0.293720	0.340869	0.459241
	30	0.236120	0.270010	0.342629	0.459946
	20	0.293341	0.339852	0.404245	0.522859
$n = 100$	$k = 100$	0.160793	0.208052	0.258440	0.339362
	90	0.137759	0.167185	0.233544	0.316399
	80	0.140670	0.168474	0.232660	0.316893
	70	0.147338	0.177223	0.238510	0.329086
	60	0.156791	0.193031	0.253804	0.342430
	50	0.173767	0.209863	0.274870	0.367155
$n = 150$	$k = 150$	0.106714	0.150029	0.193065	0.273853
	140	0.100294	0.126767	0.175397	0.251457
	130	0.099135	0.124570	0.175082	0.255210
	120	0.106236	0.136273	0.177053	0.250421
	110	0.116722	0.142494	0.190030	0.263972
	100	0.125779	0.146189	0.204310	0.282990
$n = 200$	$k = 200$	0.095888	0.120527	0.167833	0.228047
	190	0.085177	0.107770	0.150436	0.218568
	180	0.084830	0.107667	0.145002	0.215581
	170	0.091665	0.118012	0.150607	0.208230
	160	0.097845	0.110122	0.157870	0.220264
	150	0.095667	0.124148	0.163522	0.229887
$n = 250$	$k = 250$	0.078968	0.097270	0.163522	0.207971
	240	0.070861	0.089403	0.141638	0.194531
	230	0.076617	0.094512	0.126737	0.192541
	220	0.076390	0.096894	0.127134	0.194215
	210	0.084471	0.098762	0.129550	0.198693
	200	0.079550	0.105629	0.144049	0.201951
$n = 500$	$k = 500$	0.047839	0.061573	0.086172	0.147068
	490	0.047199	0.060665	0.082243	0.135416
	480	0.046720	0.061057	-	-
	470	0.049966	-	-	-
	460	-	0.063294	-	-
	450	0.053917	0.066657	-	-

12.3 Computational Results of Chapter 10

This last section contains several additional tables which give further numerical results relating to the work in Section 10.4.

Optimal Point Sets: Table 10.2 in the main text gives the approximations to the optimal point set $\mathcal{S}^*(n, 2)$ for $3 \leq n \leq 10$ returned by CMA-ES. The tables below show analogous information returned by the two other optimizers as well as the results in dimension 3.

Table 12.15: Optimal partitioning points returned by NGOpt in dimension 2.

n	Point set obtained by NGOpt
3	[0.548808, 1.055807]
4	[0.418457, 0.882434, 1.007727]
5	[0.391263, 0.620868, 0.956307, 1.037225]
6	[0.347313, 0.560015, 0.857877, 0.882838, 1.118863]
7	[0.347991, 0.489339, 0.669577, 0.908221, 0.983918, 1.036891]
8	[0.316077, 0.478213, 0.570470, 0.852702, 0.907686, 0.932709 1.067521]
9	[0.285198, 0.470163, 0.568636, 0.610603, 0.871406, 0.926506, 1.012885, 1.141138]
10	[0.272273, 0.443108, 0.532019, 0.631543, 0.819876, 0.864991, 0.925215, 1.008189, 1.132356]
15	[0.19888, 0.3777, 0.444808, 0.543373, 0.546997, 0.575153, 0.63604, 0.690127, 0.892686, 0.909171, 1.058866, 1.128335, 1.159531, 1.271836]
20	[0.223799, 0.287984, 0.324687, 0.485897, 0.499846, 0.54175, 0.57198, 0.589029, 0.615946, 0.7925, 0.817158, 0.858417, 0.908158, 0.938206, 0.960501, 0.982925, 1.101553, 1.159329, 1.319433]

Table 12.16: Optimal partitioning points returned by (1+1)-ES in dimension 2.

n	Point set obtained by (1+1)-ES
3	[0.385772, 1.414214]
4	[0.361702, 0.612305, 1.414214]
5	[0.417516, 0.605697, 0.93214, 1.135903]
6	[0.372617, 0.566202, 0.858476, 0.872371, 1.057533]
7	[0.361215, 0.476081, 0.69005, 0.956802, 0.961292, 0.997134]
8	[0.331794, 0.448127, 0.615812, 0.762824, 0.936691, 0.948517, 1.194328]
9	[0.272122, 0.50579, 0.579889, 0.631461, 0.841224, 0.872, 1.029753, 1.189806]
10	[0.300137, 0.411182, 0.477082, 0.713986, 0.775226, 0.810514, 0.878079, 1.002694, 1.203619]
15	[0.190774, 0.388459, 0.388733, 0.455958, 0.596142, 0.601442, 0.606818, 0.828665, 0.83254, 0.873472, 1.012516, 1.022808, 1.121645, 1.174104]
20	[0.249813, 0.325112, 0.359365, 0.461931, 0.483501, 0.574615, 0.5808, 0.584061, 0.606904, 0.75478, 0.84873, 0.911128, 0.916237, 0.93133, 0.942195, 0.967771, 1.009936, 1.137893, 1.363866]

Table 12.17: Optimal partitioning points returned by NGOpt in dimension 3.

n	Point set obtained by NGOpt
3	[0.795798, 1.286294]
4	[0.640472, 1.117247, 1.310707]
5	[0.566418, 1.047576, 1.083171, 1.150805]
6	[0.489396, 0.98104, 1.013217, 1.048558, 1.087265]
7	[0.448206, 0.878266, 0.91581, 0.982795, 1.063735, 1.125645]
8	[0.42716, 0.800013, 0.863311, 0.910579, 1.006799, 1.11747, 1.242571]
9	[0.408185, 0.77803, 0.830785, 0.882296, 0.890503, 1.011247, 1.169372, 1.43053]
10	[0.399794, 0.603748, 0.89242, 0.913219, 0.926596, 0.926902, 0.967523, 1.16262, 1.287299]
15	[0.351117, 0.478768, 0.688796, 0.796136, 0.821815, 0.862973, 0.879195, 0.90137, 0.9374, 0.946849, 1.045653, 1.094707, 1.186683, 1.571684]
20	[0.34235, 0.389883, 0.57646, 0.680487, 0.762054, 0.793333, 0.822658, 0.827348, 0.886069, 0.899673, 0.902049, 0.906803, 0.948087, 0.955605, 1.031147, 1.078856, 1.323314, 1.470775, 1.546017]

Table 12.18: Optimal partitioning points returned by (1+1)-ES in dimension 3.

n	Point set obtained by (1+1)-ES
3	[0.828487, 1.184358]
4	[0.688084, 0.995426, 1.23835]
5	[0.556585, 0.958897, 1.08893, 1.166941]
6	[0.504186, 0.890861, 1.01977, 1.070676, 1.220425]
7	[0.483307, 0.846026, 0.928213, 1.011243, 1.053627, 1.264038]
8	[0.438463, 0.755186, 0.855466, 0.965389, 0.971452, 1.108291, 1.364404]
9	[0.432133, 0.683787, 0.857462, 0.900443, 0.985954, 0.99401, 1.036843, 1.513559]
10	[0.433714, 0.571719, 0.82941, 0.873472, 0.954025, 0.983681, 1.123199, 1.166486, 1.190781]
15	[0.320788, 0.546152, 0.656237, 0.732797, 0.733517, 0.851861, 0.859568, 0.958379, 0.997855, 1.003527, 1.044023, 1.112383, 1.3169, 1.538414]
20	[0.308744, 0.45028, 0.612212, 0.618609, 0.652708, 0.769147, 0.791116, 0.831432, 0.852191, 0.877652, 0.927507, 0.950297, 0.997164, 1.051177, 1.070492, 1.172144, 1.212012, 1.531995, 1.640873]

Exact Discrepancy Values: Table 10.1 in the main text gives a percentage comparison of the expected L_2 discrepancies of the various point sets. For completeness sake and as a final attachment, Table 12.20 gives the exact expected discrepancy of the point same point sets.

Table 12.19: Optimal partitioning points returned by CMA-ES in dimension 3.

n	Point set obtained by CMA-ES
3	[0.852603, 1.229802]
4	[0.649511, 1.103493, 1.233462]
5	[0.557872, 0.993939, 1.111959, 1.191173]
6	[0.47672, 0.960814, 1.033193, 1.088396, 1.110393]
7	[0.450648, 0.878901, 0.905647, 1.027906, 1.096725, 1.127983]
8	[0.417508, 0.804702, 0.962827, 0.972907, 0.985132, 0.995874, 1.222424]
9	[0.385481, 0.824133, 0.863861, 0.864913, 0.950659, 0.968246, 1.027493, 1.284941]
10	[0.386811, 0.644514, 0.887476, 0.897878, 0.900528, 0.936027, 0.949786, 1.220786, 1.309354]
14	[0.358027, 0.497384, 0.609137, 0.81638, 0.875619, 0.901655, 0.904081, 0.95021, 0.959341, 0.975178, 1.012549, 1.015003, 1.248424, 1.633825]
19	[0.283066, 0.444381, 0.588593, 0.67853, 0.741726, 0.756299, 0.785411, 0.793158, 0.891821, 0.901059, 0.918331, 0.928474, 0.947623, 1.036847, 1.039228, 1.125772, 1.135725, 1.294058, 1.436106]

Table 12.20: Comparison of the expected L_2 discrepancies of different point sets: uniformly distributed random points $R(n, 2)$, the equivolume partition corresponding to $\mathcal{S}(n, 2)$ and the best sets obtained by the three black-box optimizers. The discrepancy values are evaluated with 10 000 repetitions.

n	$R(n, 2)$	$\mathcal{S}(n, 2)$	CMA-ES	NGOpt	(1+1)-ES
3	0.04614	0.02917	0.02696	0.02682	0.03061
4	0.03441	0.02035	0.01883	0.01891	0.023
5	0.02774	0.01560	0.01453	0.01464	0.01472
6	0.02327	0.01272	0.01191	0.01195	0.01192
7	0.01990	0.01070	0.01007	0.01006	0.0108
8	0.01714	0.009202	0.008722	0.008668	0.008866
9	0.01547	0.008138	0.007738	0.007764	0.007829
10	0.01492	0.007266	0.006961	0.006939	0.007051
15	0.009320	0.004746	0.004668	0.004668	0.004647
20	0.006955	0.003532	0.003541	0.003558	0.003478

Table 12.21: Comparison of the expected L_2 discrepancies of different point sets: uniformly distributed random points $R(n, 3)$, the equivolume partition corresponding to $\mathcal{S}(n, 3)$ and the best sets obtained by the three black-box optimizers. The expectation is approximated with 10 000 repetitions of the L_2 discrepancy calculation.

n	$R(n, 3)$	$\mathcal{S}(n, 3)$	CMA-ES	NGOpt	(1+1)-ES
3	0.0295	0.0220	0.01960	0.01968	0.01979
4	0.0221	0.0156	0.01433	0.01431	0.01433
5	0.0176	0.0121	0.01107	0.01104	0.01113
6	0.0148	0.00997	0.008983	0.008979	0.009115
7	0.0125	0.00836	0.007634	0.007718	0.007720
8	0.011	0.00727	0.0067	0.006712	0.006741
9	0.00989	0.00640	0.005973	0.005949	0.005967
10	0.00876	0.005713	0.005352	0.005358	0.005345
15	0.00594	0.00376	0.003569	0.003561	0.003579
20	0.00442	0.0028	0.002707	0.002692	0.002708

Bibliography

[Aar49] T. van Aardenne-Ehrenfest. **On the impossibility of a just distribution**. *Proc. Koninklijke Nederlandse Akademie van Wetenschappen* 52 (1949), 734–739 (see page 13).

[AH20] Y. Akimoto and N. Hansen. **Diagonal Acceleration for Covariance Matrix Adaptation Evolution Strategies**. *Evol. Comput.* 28:3 (2020), 405–435 (see page 158).

[AH22] Y. Akimoto and N. Hansen. **CMA-ES and advanced adaptation mechanisms**. *GECCO '22: Genetic and Evolutionary Computation Conference, COmpanion Volume, ACM* (2022), 1243–1268 (see page 158).

[Ais11] C. Aistleitner. **Covering numbers, dyadic chaining and discrepancy**. *Journal of Complexity* 27 (2011), 531–540 (see page 14).

[Ata86] M. J. Atallah. **Computing the convex hull of line intersections**. *Journal of Algorithms* 7:2 (1986), 285–288. ISSN: 0196-6774. DOI: https://doi.org/10.1016/0196-6774(86)90010-6. URL: https://www.sciencedirect.com/science/article/pii/0196677486900106 (see page 122).

[Ayg+09] E. Ayguade, N. Copty, A. Duran, J. Hoeflinger, Y. Lin, F. Massaioli, X. Teruel, P. Unnikrishnan, and G. Zhang. **The Design of OpenMP Tasks**. *IEEE Transactions on Parallel and Distributed Systems* 20:3 (Mar. 2009), 404–418. ISSN: 1558-2183. DOI: 10.1109/TPDS.2008.105 (see page 144).

[BC87] J. Beck and W.L. Chen. **Irregularities of distribution**. Cambridge: Cambridge University Press, 1987 (see page 14).

[BG02] B. Beachkofski and R. Grandhi. "Improved Distributed Hypercube Sampling." In: *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, 2002, 1–7. DOI: 10.2514/6.2002-1274 (see pages 14, 20).

[BLV08] D. Bilyk, M.T. Lacey, and A. Vagharshakyan. **On the small ball inequality in all dimensions**. *J. Funct. Anal.* 254 (2008), 2470–2502. DOI: https://doi.org/10.1016/j.jfa.2007.09.010 (see pages 14, 40).

[Bou+17] O. Bousquet, S. Gelly, K. Kurach, O. Teytaud, and D. Vincent. **Critical Hyper-Parameters: No Random, No Cry**. *CoRR* abs/1706.03200 (2017). arXiv: 1706.03200 (see pages 2, 66).

[BTY12] D. Bilyk, V.N. Temlyakov, and R. Yu. **Fibonacci sets and symmetrization in discrepancy theory**. *J. Complex.* 28:1 (2012), 18–36. DOI: 10.1016/j.jco.2011.07.001 (see page 95).

[BW79] E. Braaten and G. Weller. **An Improved Low-Discrepancy Sequence for Multidimensional Quasi-Monte Carlo Integration**. *J. of Comput. Phys.* 33:2 (1979), 249–258. DOI: 10.1016/0021-9991(79)90019-6 (see page 95).

[BZ93] P. Bundschuh and Y.C. Zhu. **A method for exact calculation of the discrepancy of low-dimensional point sets**. *I. Abh. Math. Sem. univ. Hamburg* 63 (1993), 115–133 (see page 23).

[Cau+20] M.-L. Cauwet, C. Couprie, J. Dehos, P. Luc, J. Rapin, M. Rivière, F. Teytaud, O. Teytaud, and N. Usunier. **Fully Parallel Hyperparameter Search: Reshaped Space-Filling**. In: *Proc. of the 37th International Conference on Machine Learning, ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, 1338–1348. URL: http://proceedings.mlr.press/v119/cauwet20a.html (see pages 2, 66).

[CDP22] F. Clément, C. Doerr, and L. Paquete. **Star discrepancy subset selection: Problem formulation and efficient approaches for low dimensions**. *Journal of Complexity* 70 (2022), 101645. DOI: 10.1016/j.jco.2022.10164 (see pages 4, 17, 22, 66, 68).

[CDP24] F. Clément, C. Doerr, and L. Paquete. **Heuristic approaches to obtain low-discrepancy point sets via subset selection**. *Journal of Complexity* 83 (2024), 101852. ISSN: 0885-064X. DOI: https://doi.org/10.1016/j.jco.2024.101852. URL: https://www.sciencedirect.com/science/article/pii/S0885064X24000293 (see pages 4, 98).

[Cha+20] K. I. Chatzilygeroudis, A. Cully, V. Vassiliades, and J.-B. Mouret. **Quality-Diversity Optimization: a Novel Branch of Stochastic Optimization**. *CoRR* abs/2012.04322 (2020). arXiv: 2012.04322. URL: https://arxiv.org/abs/2012.04322 (see page 153).

[Cha00] B. Chazelle. **The Discrepancy method**. *Cambridge University Press (Cambridge)* (2000) (see page 11).

[Cha13] T.M. Chan. **Klee's Measure Problem Made Easy**. In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. 2013, 410–419. DOI: 10.1109/FOCS.2013.51 (see page 170).

[CKP24] F. Clément, N. Kirk, and F. Pausinger. *Monte Carlo Methods and Applications* (2024). DOI: doi:10.1515/mcma-2023-2025. URL: https://doi.org/10.1515/mcma-2023-2025 (see pages 5, 154, 158).

[Clé] F. Clément. https://github.com/frclement/ (see page 130).

[Clé+23a] F. Clément, C. Doerr, K. Klamroth, and L. Paquete. **Constructing Optimal L_∞ Star Discrepancy Sets**. *CoRR* abs/2311.17463 (2023). DOI: 10.48550/ARXIV.2311.17463. arXiv: 2311.17463. URL: https://doi.org/10.48550/arXiv.2311.17463 (see pages 4, 17, 37, 136, 167).

[Clé+23b] F. Clément, D. Vermetten, J. de Nobel, A. D. Jesus, L. Paquete, and C. Doerr. *Reproducibility files and additional figures*. https://doi.org/10.5281/zenodo.7630260. Feb. 2023 (see pages 103, 144).

[Clé+23c] F. Clément, D. Vermetten, J. de Nobel, A.D. Jesus, C. Doerr, and L. Paquete. **Computing Star Discrepancies with Numerical Black-Box Optimization Algorithms**. *Proc. of GECCO'23* (2023), 1330–1338. URL: https://dl.acm.org/doi/10.1145/3583131.3590456 (see pages 5, 143, 144).

[Clé23] F. Clément. **Extending the Kritzinger sequence: more points and higher dimensions** (2023). https://webia.lip6.fr/~fclement/fclement (see pages 5, 7, 118).

[Cor35] J.G. van der Corput. **Verteilungsfunktionen II**. *Akad. Wetensch. Amsterdam Proc.* 38 (1935), 1058–1066 (see page 18).

[CST14] William Chen, Anand Srivastav, and Giancarlo Travaglini. **A Panorama of Discrepancy Theory**. Vol. 2107. Jan. 2014. ISBN: 978-3-319-04695-2. DOI: 10.1007/978-3-319-04696-9 (see page 11).

[Cyg+15] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. **Parameterized Complexity**. Springer Publishing Company, Incorporated, 2015 (see page 24).

[Dav56] H. Davenport. **Note on irregularities of distribution**. *Mathematika* 3 (1956), 131–135 (see page 15).

[DDG18] B. Doerr, C. Doerr, and M. Gnewuch. "Probabilistic Lower Bounds for the Discrepancy of Latin Hypercube Samples." In: *Contemporary Computational Mathematics - A Celebration of the 80th Birthday of Ian Sloan*. Ed. by Josef Dick, Frances Y. Kuo, and Henryk Woźniakowski. Springer, 2018, 339–350. ISBN: 978-3-319-72456-0. DOI: 10.1007/978-3-319-72456-0_16 (see page 14).

[DEM96] D.P. Dobkin, D. Eppstein, and D.P. Mitchell. **Computing the Discrepancy with Applications to Supersampling Patterns**. *ACM Trans. Graph.* 15:4 (1996), 354–376. DOI: 10.1145/234535.234536 (see pages 8, 25, 82, 144).

[DGS05] B. Doerr, M. Gnewuch, and A. Srivastav. **Bounds and constructions for the star-discrepancy via δ-covers**. *Journal of Complexity* 21:5 (2005), 691–709. ISSN: 0885-064X. DOI: https://doi.org/10.1016/j.jco.2005.05.002 (see pages 15, 28).

[DGW14] C. Doerr, M. Gnewuch, and M. Wahlström. **Calculation of Discrepancy Measures and Applications in: W. Chen, A. Srivastav, G. Travaglini (Eds.)** *A Panorama of Discrepancy Theory, Springer* (2014), 621–678 (see pages 11, 17, 18, 21, 25, 59).

[Doe+08] B. Doerr, M. Gnewuch, P. Kritzer, and F. Pillichshammer. **Component-by-component construction of low-discrepancy point sets of small size**. *Monte Carlo Methods and Applications* 14:2 (2008), 129–149. DOI: doi:10.1515/MCMA.2008.007. URL: https://doi.org/10.1515/MCMA.2008.007 (see page 15).

[Doe+18] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck. **IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics**. *CoRR* abs/1810.05281 (2018). Available at http://arxiv.org/abs/1810.05281. A more up-to-date documentation of IOHprofiler is available at https://iohprofiler.github.io/ (see page 149).

[Doe14] B. Doerr. **A lower bound for the discrepancy of a random point set**. *Journal of Complexity* 30:1 (2014), 16–20. DOI: 10.1016/j.jco.2013.06.001 (see pages 14, 15).

[Doe22] B. Doerr. **A sharp discrepancy bound for jittered sampling**. *Math. Comput.* 91:336 (2022), 1871–1892 (see page 14).

[DP10] J. Dick and F. Pillichshammer. **Digital Nets and Sequences**. Cambridge: Cambridge University Press, 2010 (see pages 2, 17).

[DP14] J. Dick and F. Pillichshammer. **Discrepancy Theory and Quasi-Monte Carlo Integration in: W. Chen, A. Srivastav, G. Travaglini (Eds.)** *A Panorama of Discrepancy Theory, Springer* (2014), 539–620 (see page 32).

[DR13] C. Doerr and F.-M. de Rainville. **Constructing low star discrepancy point sets with genetic algorithms**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2013, 789–796 (see pages 16, 28, 30, 65, 110).

[DS90] G. Dueck and T. Scheuer. **Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing**. *J. Comput. Phys.* 90 (1990), 161–175 (see page 27).

[Dud78] R.M. Dudley. **Central Limit Theorems for Empirical Measures**. *The Annals of Probability* 6:6 (1978), 899–929. DOI: 10.1214/aop/1176995384. URL: https://doi.org/10.1214/aop/1176995384 (see page 28).

[Dwi+19] R. Dwivedi, O.N. Feldheim, O. Gurel-Gurevich, and A. Ramdas. **The power of online thinning in reducing discrepancy**. *Probability Theory and Related Fields* 174 (2019), 103–131 (see pages 66, 67).

[ET48a] P. Erdős and P. Turán. **On a problem in the theory of uniform distribution I**. *Nederl. Akad. Wetensch.* 51 (1948), 1146–1154 (see pages 16, 17).

[ET48b] P. Erdős and P. Turán. **On a problem in the theory of uniform distribution II**. *Nederl. Akad. Wetensch.* 51 (1948), 1262–1269 (see pages 16, 17).

[Fau82] H. Faure. **Discrepancy of sequences associated with a number system (in dimension s)**. *Acta. Arith* 41:4 (1982). In French., 337–351 (see page 18).

[Fox86] B. Fox. **Algorithm 647:Implementation and Relative Efficiency of Quasirandom Sequence Generators**. *ACM Transactions on Mathematical Software* 12:4 (1986), 362–376 (see page 147).

[GH21] M. Gnewuch and N. Hebbinghaus. **Discrepancy bounds for a class of negatively dependent random points including Latin hypercube samples**. *Annals of Applied Probability* (2021). To appear. Available at https://imstat.org/journals-and-publications/annals-of-applied-probability/annals-of-applied-probability-future-papers/ (see pages 14, 15).

[Gia+12] P. Giannopoulos, C. Knauer, M. Wahlström, and D. Werner. **Hardness of discrepancy computation and ϵ-net verification in high dimension**. *Journal of Complexity* 28 (2012), 162–176 (see page 24).

[GJ90] M.R. Garey and D.S. Johnson. **Computers and Intractability; A Guide to the Theory of NP-Completeness**. USA: W. H. Freeman & Co., 1990. ISBN: 0716710455 (see page 69).

[GJ97] S. Galanti and A. Jung. **Low-discrepancy sequences: Monte-Carlo simulation of option prices**. *J. Deriv* (1997), 63–83 (see page 2).

[GLM22] P. Gilibert, T. Lachmann, and C. Müllner. **The VC-dimension of axis-parallel boxes on the torus**. *Journal of Complexity* 68 (2022), 101600 (see page 61).

[Gne08] M. Gnewuch. **Bracketing numbers for axis-parallel boxes and applications to geometric discrepancy**. *Journal of Complexity* 24:2 (2008), 154–172. ISSN: 0885-064X. DOI: https://doi.org/10.1016/j.jco.2007.08.003 (see pages 15, 115).

[Gne12] M. Gnewuch. **Entropy, Randomization, Derandomization, and Discrepancy**. In: *Monte Carlo and Quasi-Monte Carlo Methods 2010*. Ed. by Leszek Plaskota and Henryk Woźniakowski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 43–78. ISBN: 978-3-642-27440-4 (see page 28).

[Gne24] M. Gnewuch. **Improved bounds for the bracketing number of orthants or revisiting an algorithm of Thiémard to compute bounds for the star discrepancy** (2024). URL: https://arxiv.org/abs/2401.00801 (see page 28).

[GPW20] M. Gnewuch, H. Pasing, and C. Weiß. **A generalized Faulhaber inequality, improved bracketing covers and applications to discrepancy**. *Mathematics of Computation* 90 (2020), 2873–2898 (see pages 14, 15).

[GSW09] M. Gnewuch, A. Srivastav, and C. Winzen. **Finding optimal volume subintervals with k points and calculating the star discrepancy are NP-hard problems**. *J. Complexity* 25:2 (2009), 115–127 (see pages 24, 29, 67, 69, 70).

[Gur23] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. https://www.gurobi.com. 2023 (see pages 37, 137).

[GWW12] M. Gnewuch, M. Wahlström, and C. Winzen. **A New Randomized Algorithm to Approximate the Star Discrepancy Based on Threshold Accepting**. *SIAM J. Numerical Analysis* 50:2 (2012), 781–807. DOI: 10.1137/110833865 (see pages 25–27, 144, 153).

[Hal60] J.H. Halton. **On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals**. *Numerische Mathematik* 2 (1960), 84–90 (see page 147).

[Hal64] J.H. Halton. **Algorithm 247: Radical-Inverse Quasi-random Point Sequence**. *Communications of the ACM* 7:12 (1964), 701–702 (see page 18).

[Han16] N. Hansen. **The CMA Evolution Strategy: A Tutorial** (2016). URL: https://arxiv.org/abs/1604.00772 (see page 158).

[Hei+01] S. Heinrich, E. Novak, G. Wasilkowski, and H. Wozniakowski. **The inverse of the star-discrepancy depends linearly on the dimension**. *Acta Arithmetica* 96 (Jan. 2001), 279–302. DOI: 10.4064/aa96-3-7 (see pages 14, 15).

[Hei96] S. Heinrich. **Efficient algorithms for computing the L_2 discrepancy**. *Math. Comp* 65 (1996), 1621–1633 (see page 31).

[Hin04] A. Hinrichs. **Covering numbers, Vapnik-Červonenkis classes and bounds for the star-discrepancy**. *J. Complexity* 20 (2004), 477–483 (see page 15).

[Hin13] A. Hinrichs. **Discrepancy, Integration and Tractability**. In: *Monte Carlo and Quasi-Monte Carlo Methods 2012*. Ed. by Josef Dick, Frances Y. Kuo, Gareth W. Peters, and Ian H. Sloan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, 129–172. ISBN: 978-3-642-41095-6 (see pages 30, 110).

[Hla61] E. Hlawka. **Funktionen von beschränkter Variation in der Theorie der Gleichverteilung**. *Ann. Mat. Pum Appl.* 54 (1961), 325–333 (see pages iii, 2).

[Hla62] E. Hlawka. **Zur angenäherten Berechnung mehrfacher Integrale**. *Monatshefte Math* 66 (1962), 140–151 (see page 18).

[HO01] N. Hansen and A. Ostermeier. **Completely Derandomized Self-Adaptation in Evolution Strategies**. *Evolutionary Computation* 9:2 (2001), 159–195. DOI: 10.1162/106365601750190398. URL: https://doi.org/10.1162/106365601750190398 (see pages 148, 158).

[HO14] A. Hinrichs and J. Oettershagen. **Optimal Point Sets for Quasi-Monte Carlo Integration of Bivariate Periodic Functions with Bounded Mixed Derivatives**. In: *Monte Carlo and Quasi-Monte Carlo Methods*. 2014 (see pages 14, 16, 53).

[HO96] N. Hansen and A. Ostermeier. **Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation**. *Proceedings of 1996 IEEE International Conference on Evolutionary Computation* (1996), 312–317 (see page 158).

[JK08] S. Joe and F.Y. Kuo. **Constructing Sobol Sequences with Better Two-Dimensional Projections**. *SIAM Journal on Scientific Computing* 30:5 (2008), 2635–2654. DOI: 10.1137/070709359. URL: https://doi.org/10.1137/070709359 (see page 18).

[Joe12] S. Joe. **An Intermediate Bound on the Star Discrepancy**. *Springer Proceedings in Mathematics and Statistics* 23 (Jan. 2012). DOI: 10.1007/978-3-642-27440-4_25 (see page 17).

[KE95] J. Kennedy and R. Eberhart. **Particle swarm optimization**. In: *Proc. of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948. DOI: 10.1109/ICNN.1995.488968 (see page 149).

[KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. **Optimization by Simulated Annealing**. *Science* 220 (1983), 671–680 (see page 27).

[Koc+14] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang. **The unconstrained binary quadratic programming problem: A survey**. *Journal of Combinatorial Optimization* 28 (July 2014). DOI: 10.1007/s10878-014-9734-0 (see pages 96, 97).

[Kok43] J.F. Koksma. **A general theorem from the theory of the uniform distribution modulo 1**. *Mathematica B (Zutphen)* 1 (1942/1943), 7–11 (see pages iii, 2).

[Kok50] J. F. Koksma. **Some theorems on diophantine inequalities**. *Math. Centrum Amsterdam Scriptum* 5 (1950) (see page 16).

[Kor59] N. M. Korobov. **The approximate computation of multiple integrals**. *Dokl. Akad. Nauk SSSR* 124 (1959), 1207–1210 (see page 18).

[KP21] M. Kiderlen and F. Pausinger. **Discrepancy of stratified samples from partitions of the unit cube**. *Monatshefte für Mathematik* 195 (2021), 267–306 (see pages 9, 155, 157, 158, 162).

[KP22] M. Kiderlen and F. Pausinger. **On a partition with a lower expected L2-discrepancy than classical jittered sampling**. *Journal of Complexity* 70 (2022), 101616. ISSN: 0885-064X. DOI: https://doi.org/10.1016/j.jco.2021.101616. URL: https://www.sciencedirect.com/science/article/pii/S0885064X21000716 (see pages 9, 158).

[Kri22] R. Kritzinger. **Uniformly distributed sequences generated by a greedy minimization of the L_2 discrepancy**. *Moscow Journal of Combinatorics and Number Theory* 11:3 (2022). https://arxiv.org/abs/2109.06298, 215–236 (see pages 4, 8, 32, 33, 118, 129–131, 141).

[LEc+22] P. L'Ecuyer, P. Marion, M. Godin, and F. Puchhammer. **A Tool for Custom Construction of QMC and RQMC Point Sets**. *Monte Carlo and Quasi-Monte Carlo Methods* (2022), 451–470. DOI: https://link.springer.com/chapter/10.1007/978-3-030-98319-2_3 (see page 169).

[Lev96] V.F. Lev. **Translations of nets and relationship between supreme and L^k discrepancies**. *Acta Math. Hung.* 12 (1996), 1–12 (see page 61).

[LL01] P. Larrañaga and J.A. Lozano. **Estimation of distribution algorithms: A new tool for evolutionary computation**. Vol. 2. Springer Science & Business Media, 2001 (see page 148).

[LP07] G. Larcher and F. Pillichshammer. **A note on optimal point distributions in $[0,1)^s$**. *Journal of Computational and Applied Mathematics* 206:2 (2007), 977–985. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2006.09.004. URL: https://www.sciencedirect.com/science/article/pii/S0377042706005620 (see page 15).

[LP16] G. Larcher and F. Puchhammer. **An Improved Bound for the Star Discrepancy of Sequences in the Unit Interval**. *Uniform Distribution Theory* 11 (2016), 1–14 (see page 13).

[Mat10] J. Matoušek. **Geometric Discrepancy**. *2nd edition, Springer Berlin* (2010) (see pages 11, 12, 53).

[Mat98] J. Matoušek. **On the L2-discrepancy for anchored boxes**. *J. Complexity* 14 (1998), 527–556 (see page 32).

[MBC79] M.D. McKay, R.J. Beckman, and W.J. Conover. **A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code**. *Technometrics* 21 (1979), 239–245. ISSN: 00401706 (see page 14).

[MC94] W.J. Morokoff and R.E. Caflisch. **Quasi-Random Sequences and Their Discrepancies**. *SIAM J. Sci. Comput.* 15 (1994), 1251–1279. URL: https://api.semanticscholar.org/CorpusID:5051555 (see page 59).

[Meu+21] L. Meunier, H. Rakotoarison, P. K. Wong, B. Roziere, J. Rapin, O. Teytaud, A. Moreau, and C. Doerr. **Black-box optimization revisited: Improving algorithm selection wizards through massive benchmarking**. *IEEE Transactions on Evolutionary Computation* 26:3 (2021), 490–500 (see pages 148, 159).

[MML14] R. Martins, V. Manquinho, and I. Lynce. **Open-WBO: A modular MaxSAT solver**. *Theory and Applications of Satisfiability Testing- SAT* (2014), 438–445 (see page 80).

[MMM04] H. Maaranen, K. Miettinen, and M.M. Mäkelä. **Quasi-random initial population for genetic algorithms**. *Computers & Mathematics with Applications* 47:12 (2004), 1885–1895. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2003.07.011. URL: https://www.sciencedirect.com/science/article/pii/S0898122104840240 (see page 2).

[Neu+18] A. Neumann, W. Gao, C. Doerr, F. Neumann, and M. Wagner. **Discrepancy-based evolutionary diversity optimization**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'18)*. ACM, 2018, 991–998. DOI: 10.1145/3205455.3205532 (see pages 6, 66).

[Nie72] H. Niederreiter. **Discrepancy and Convex Programming**. *Ann. Mat. Pura Appl.* 93 (1972), 89–97 (see pages 13, 21, 71).

[Nie92] H. Niederreiter. **Random Number Generation and Quasi-Monte Carlo Methods**. Vol. 63. SIAM CBMS-NSF Regional Conference Series in Applied Mathematics. Philadelphia: SIAM, 1992 (see pages 14, 17–19, 59).

[Nob+24] J. de Nobel, F. Ye, D. Vermetten, H. Wang, C. Doerr, and T. Bäck. **IOHexperimenter: Benchmarking Platform for Iterative Optimization Heuristics**. *Evolutionary Computation* (2024). DOI: https://doi.org/10.1162/evco_a_00342 (see pages 147, 162).

[NW10] E. Novak and H. Woźniakowski. **Tractability of Multivariate problems, Volume 2**. *Eur. Math. Soc. Publ. House* (2010) (see pages 15, 16, 30, 99, 110).

[Ost09] V. Ostromoukhov. **Recent Progress in Improvement of Extreme Discrepancy and Star Discrepancy of One-dimensional Sequences**. *Monte Carlo and Quasi-Monte Carlo Methods 2008* (2009), 561–572 (see pages 13, 18, 131).

[Owe23] Art B. Owen. **Practical Quasi-Monte Carlo Integration**. https://artowen.su.domains/mc/practicalqmc.pdf, 2023 (see page 38).

[Pau+22] L. Paulin, N. Bonneel, D. Coeurjoly, J.-C. Iehl, A. Keller, and V. Ostromoukhov. **MatBuilder: Mastering Sampling Uniformiy over projections**. *ACM Transactions on Graphics (proceedings of SIGGRAPH)* (2022) (see page 2).

[Pau19] F. Pausinger. **On the Intriguing Search for Good Permutations**. *Uniform distribution theory* 14:1 (2019), 53–86. DOI: doi:10.2478/udt-2019-0005. URL: https://doi.org/10.2478/udt-2019-0005 (see page 18).

[Pau21] F. Pausinger. **Greedy energy minimization can count in binary: point charges and the van der Corput sequence**. *Annali di Matematica* 200 (2021), 165–186 (see page 33).

[Pau23] F. Pausinger. **On the expected $\mathcal{L}_2$-discrepancy of stratified samples from parallel lines** (2023). URL: https://arxiv.org/pdf/2310.13927.pdf (see pages 154, 162).

[Pow94] M. J. D. Powell. "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation." In: *Advances in Optimization and Numerical Analysis*. Ed. by S. Gomez and J.-P. Hennart. Dordrecht: Springer Netherlands, 1994, 51–67. ISBN: 978-94-015-8330-5. DOI: 10.1007/978-94-015-8330-5_4. URL: https://doi.org/10.1007/978-94-015-8330-5_4 (see page 148).

[PS16] F. Pausinger and S. Steinerberger. **On the discrepancy of jittered sampling**. *Journal of Complexity* 33 (2016), 199–216 (see page 155).

[PSS16] J.K. Pugh, L.B. Soros, and K.O. Stanley. **Quality Diversity: A New Frontier for Evolutionary Computation**. *Frontiers Robotics AI* 3 (2016), 40. DOI: 10.3389/frobt.2016.00040. URL: https://doi.org/10.3389/frobt.2016.00040 (see page 153).

[Pug+15] J.K. Pugh, L.B. Soros, P.A. Szerlip, and K.O. Stanley. **Confronting the Challenge of Quality Diversity**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2015, 967–974. DOI: 10.1145/2739480.2754664. URL: https://doi.org/10.1145/2739480.2754664 (see page 153).

[PVC06] T. Pillards, B. Vandewoestyne, and R. Cools. **Minimizing the L_2 and L_∞ star discrepancies of a single point in the unit hypercube**. *Journal of Computational and Applied Mathematics* 197 (Dec. 2006), 282–285. DOI: 10.1016/j.cam.2005.11.005 (see page 15).

[Rij+16] S. van Rijn, H. Wang, M. van Leeuwen, and T. Bäck. **Evolving the structure of Evolution Strategies**. *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016,* (2016) (see page 158).

[Rot54] K.F. Roth. **On irregularities of distribution**. *Mathematika* 1:2 (1954), 73–79. DOI: https://doi.org/10.1112/S0025579300000541 (see pages 13, 14).

[Rot79] K.F. Roth. **On irregularities of distribution III**. *Acta Arith.* 35 (1979), 373–384 (see page 15).

[RT18] J. Rapin and O. Teytaud. *Nevergrad - A gradient-free optimization platform*. https://GitHub.com/FacebookResearch/Nevergrad. 2018 (see pages 148, 162).

[SC02] M. Skriganov and W. Chen. **Explicit constructions in the classical mean squares problem in irregularities of point distribution**. *Journal für die reine und angewandte Mathematik* 2002:545 (2002), 67–95. DOI: doi:10.1515/crll.2002.037. URL: https://doi.org/10.1515/crll.2002.037 (see page 15).

[Sch72] W.M. Schmidt. **Irregularities of distribution VII**. *Acta. Arith* 21 (1972), 45–50 (see page 13).

[Sha10] M. Shah. *Monte Carlo Methods and Applications* 16:3-4 (2010), 379–398. DOI: doi:10.1515/mcma.2010.014. URL: https://doi.org/10.1515/mcma.2010.014 (see page 29).

[Shi12] O. Shir. "Niching in Evolutionary Algorithms." In: *Handbook of Natural Computing*. Ed. by Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok. Springer, 2012, 1035–1069. ISBN: 978-3-540-92910-9. DOI: 10.1007/978-3-540-92910-9_32. URL: https://doi.org/10.1007/978-3-540-92910-9_32 (see page 153).

[SJ94] I.H. Sloan and S. Joe. **Lattice Methods for Numerical Integration**. *Clarendon Press, Oxford* (1994) (see page 17).

[Sob67] I.M. Sobol. **On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals**. *USSR Computational Mathematics and Mathematical Physics* 7:4 (1967), 86–112 (see pages 17, 147).

[SP97] R. Storn and K. Price. **Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces**. *Journal of Global Optimization* 11:4 (1997), 341–359. ISSN: 0925-5001. DOI: 10.1023/A:1008202821328. URL: https://doi.org/10.1023/A:1008202821328 (see page 148).

[Spa92] J.C. Spall. **Multivariate stochastic approximation using a simultaneous perturbation gradient approximation**. *IEEE Transactions on Automatic Control* 37:3 (1992), 332–341 (see page 149).

[SS68] M. Schumer and K. Steiglitz. **Adaptive step size random search**. *IEEE Transactions on Automatic Control* 13 (1968), 270–276 (see page 160).

[Ste19] S. Steinerberger. **A non-local functional promoting low-discrepancy point sets**. *Journal of Complexity* 54 (2019), 101410 (see pages 7, 17, 32, 33, 98, 110, 130).

[Ste20] S. Steinerberger. **Dynamically defined sequences with small discrepancy**. *Monatshefte Math* 191 (2020), 639–655 (see page 32).

[Ste24] S. Steinerberger. **On combinatorial properties of greedy Wasserstein minimization**. *Journal of Mathematical Analysis and Applications* 532:1 (2024), 127940. ISSN: 0022-247X. DOI: https://doi.org/10.1016/j.jmaa.2023.127940. URL: https://www.sciencedirect.com/science/article/pii/S0022247X23009435 (see pages 8, 35, 118, 129, 130, 171).

[SW98] I.H. Sloan and H. Woźniakowski. **When Are Quasi-Monte Carlo Algorithms Efficient for High Dimensional Integrals?** *Journal of Complexity* 14:1 (1998), 1–33. ISSN: 0885-064X. DOI: https://doi.org/10.1006/jcom.1997.0463. URL: https://www.sciencedirect.com/science/article/pii/S0885064X97904635 (see page 169).

[SWN03] T. J. Santner, B.J. Williams, and W. I. Notz. **The Design and Analysis of Computer Experiments**. Springer Series in Statistics. Springer, 2003. ISBN: 978-1-4419-2992-1. DOI: 10.1007/978-1-4757-3799-8 (see page 2).

[Thi00] E. Thiémard. **Sur le calcul et la majoration de la discrépance à l'origine**. *PhD thesis École polytechnique fédérale de Lausanne EPFL, nbr 2259* (2000). URL: https://infoscience.epfl.ch/record/32735 (see pages 28, 29).

[Thi01a] E. Thiémard. **An algorithm to compute bounds for the star discrepancy**. *Journal of Complexity* 17 (2001), 850–880 (see pages 28, 115).

[Thi01b] E. Thiémard. **Optimal volume subintervals with k points and star discrepancy via integer programming**. *Math. Meth. Oper. Res.* 54 (2001), 21–45 (see pages 28, 29, 65).

[VC06] B. Vandewoestyne and R. Cools. **Good permutations for deterministic scrambled Halton sequences in terms of L_2-discrepancy**. *Journal of Computational and Applied Mathematics* 189 (2006), 341–361. DOI: https://doi.org/10.1016/j.cam.2005.05.022 (see pages 18, 67).

[Wan+22] H. Wang, D. Vermetten, F. Ye, C. Doerr, and T. Bäck. **IOHanalyzer: Detailed Performance Analyses for Iterative Optimization Heuristics**. *ACM Trans. Evol. Learn. Optim.* 2 (2022). ISSN: 2688-299X. DOI: 10.1145/3510426. URL: https://doi.org/10.1145/3510426 (see page 149).

[War72] T.T. Warnock. **Computational inverstigations of low-discrepancy point sets**. *in Applications of number theory to numerical analysis, ed. by S.K. Zaremba (Acedemic Press, New York)* (1972) (see page 31).

[Wey16] H. Weyl. **Über die Gleichverteilung von Zahlen mod. Eins.** *Math. Ann.* 77 (1916), 313–352 (see page 1).

[WF97] P. Winker and K.T. Fang. **Applications of Threshold-Accepting to the evaluation of the discrepancy of a set of points**. *SIAM J. Numerical Analysis* 34 (1997), 2028–2042 (see pages 23, 27).

[Whi77] B.E. White. **On optimal extreme-discrepancy point sets in the square**. *Numer. Math.* 27 (1976/1977), 157–164 (see pages 5, 15, 37–39, 43).