

A NEW RANDOMIZED ALGORITHM TO APPROXIMATE THE STAR DISCREPANCY BASED ON THRESHOLD ACCEPTING

MICHAEL GNEWUCH*, MAGNUS WAHLSTRÖM†, AND CAROLA WINZEN†

Abstract. We present a new algorithm for estimating the star discrepancy of arbitrary point sets. Similar to the algorithm for discrepancy approximation of Winker and Fang [SIAM J. Numer. Anal. 34 (1997), 2028–2042] it is based on the optimization algorithm threshold accepting. Our improvements include, amongst others, a non-uniform sampling strategy which is more suited for higher-dimensional inputs and additionally takes into account the topological characteristics of given point sets, and rounding steps which transform axis-parallel boxes, on which the discrepancy is to be tested, into *critical test boxes*. These critical test boxes provably yield higher discrepancy values, and contain the box that exhibits the maximum value of the local discrepancy. We provide comprehensive experiments to test the new algorithm. Our randomized algorithm computes the exact discrepancy frequently in all cases where this can be checked (i.e., where the exact discrepancy of the point set can be computed in feasible time). Most importantly, in higher dimension the new method behaves clearly better than all previously known methods.

Key words. computational geometry, discrepancy, integer programming, numerical integration, optimization heuristics, threshold-accepting

AMS subject classifications. 11K38, 65K10, 11Y16

1. Introduction. Discrepancy theory analyzes the irregularity of point distributions and has considerable theoretical and practical relevance. There are many different discrepancy notions with a wide range of applications as in optimization, combinatorics, pseudo random number generation, option pricing, computer graphics, and other areas, see, e.g., the monographs [2, 5, 7, 12, 14, 30, 31, 33, 34].

In particular for the important task of multivariate or infinite dimensional numerical integration, which arises frequently in fields such as finance, statistics, physics or quantum chemistry, quasi-Monte Carlo algorithms relying on low-discrepancy samples have been studied extensively in the last decades. For several classes of integrands the error of quasi-Monte Carlo approximation can be expressed in terms of the discrepancy of the set of sample points. This is put into a quantitative form by inequalities of Koksma-Hlawka- or Zaremba-type, see, e.g., [7, 19, 34] and the literature mentioned therein. The essential point here is that a set of sample points with small discrepancy results in a small integration error. The most important and most extensively studied discrepancy notion is the *star discrepancy*, which is defined as follows.

Let $X = (x^i)_{i=1}^n$ be a finite sequence in the d -dimensional (half-open) unit cube $[0, 1]^d$. For $y = (y_1, \dots, y_d) \in [0, 1]^d$ let $A(y, X)$ be the number of points of X lying in the d -dimensional half-open subinterval $[0, y) := [0, y_1) \times \dots \times [0, y_d)$, and let V_y be the d -dimensional (Lebesgue) volume of $[0, y)$. We call

$$\delta(y) = \delta(y, X) := V_y - \frac{1}{n} A(y, X)$$

the *local discrepancy* of X in the subinterval $[0, y)$, and the supremum norm of the local discrepancy,

$$d_\infty^*(X) := \sup_{y \in [0, 1]^d} |\delta(y, X)|,$$

*Institut für Informatik, Christian-Albrechts-Universität Kiel, Christian-Albrechts-Platz 4, 24098 Kiel, Germany.

†Max-Planck-Institut für Informatik, Campus E1 4, 66123 Saarbrücken, Germany.

the L^∞ -star discrepancy, or simply the *star discrepancy* of X .

A generalization of the star discrepancy which attracted more and more attention over the last few years, is the so-called *weighted star discrepancy*, see, e.g., [6, 24, 25, 37]. In particular, it is very promising for finance applications, see [38]. In this article we focus on algorithms to approximate the star discrepancy, but the reader should keep in mind that these build the necessary basis for algorithms to approximate the weighted star discrepancy.

In many applications it is of interest to measure the quality of certain sets by calculating their star discrepancy, e.g., to test whether successive pseudo random numbers are statistically independent [33], or whether given sample sets are suitable for multivariate numerical integration of certain classes of integrands. As explained in [9], the fast calculation or approximation of the star discrepancy would moreover allow efficient randomized semi-constructions of low-discrepancy samples of moderate size (meaning at most polynomial in the dimension d). Actually, there are derandomized algorithms known to construct such samples deterministically [9, 10, 11], but these exhibit high running times. Therefore, efficient semi-constructions would be helpful to avoid the costly derandomization procedures. The critical step in the semi-construction is the efficient calculation (or approximation) of the discrepancy of a given point set.

The L^2 -star discrepancy (i.e., the L^2 -norm of the local discrepancy function) of a given n -point set in dimension d can be computed with the help of Warnock's formula [42] with $O(dn^2)$ arithmetic operations. Heinrich and Frank provided an asymptotically faster algorithm using $O(n(\log n)^{d-1})$ operations for fixed d [16, 23]. Similarly efficient algorithms are not known for the star discrepancy. In fact it is known that the problem of calculating the star discrepancy of arbitrary point sets is an NP -hard problem [20]. Furthermore, it was shown recently that it is also a $W[1]$ -hard problem with respect to the parameter d [17]. So it is not very surprising that all known algorithms for calculating the star discrepancy or approximating it up to a user-specified error exhibit running times exponential in d , see [8, 18, 40, 41]. Let us have a closer look at the problem: For a finite sequence $X = (x^i)_{i=1}^n$ in $[0, 1]^d$ and for $j \in \{1, \dots, d\}$ we define

$$\Gamma_j(X) = \{x_j^i \mid i \in \{1, \dots, n\}\} \quad \text{and} \quad \bar{\Gamma}_j(X) = \Gamma_j(X) \cup \{1\},$$

and the grids

$$\Gamma(X) = \Gamma_1(X) \times \dots \times \Gamma_d(X) \quad \text{and} \quad \bar{\Gamma}(X) = \bar{\Gamma}_1(X) \times \dots \times \bar{\Gamma}_d(X).$$

Then we obtain

$$d_\infty^*(X) = \max \left\{ \max_{y \in \Gamma(X)} \left(V_y - \frac{1}{n} A(y, X) \right), \max_{y \in \bar{\Gamma}(X)} \left(\frac{1}{n} \bar{A}(y, X) - V_y \right) \right\}, \quad (1.1)$$

where $\bar{A}(y, X)$ denotes the number of points of X lying in the closed d -dimensional subinterval $[0, y]$. (For a proof see [20] or [32, Thm. 2].) Thus, an enumeration algorithm would provide us with the exact value of $d_\infty^*(X)$. But since the cardinality of the grid $\Gamma(X)$ for almost all X is n^d , such an algorithm would be infeasible for large values of n and d .

Since an efficient algorithm for the exact calculation of the star discrepancy is unlikely, and since no efficient approximation algorithm is known, other authors tried to deal with this problem by using optimization heuristics. In [43], Winker and Fang

used threshold accepting to find lower bounds for the star discrepancy. Threshold accepting [13] is a refined randomized local search algorithm based on a similar idea as the simulated annealing algorithm [28]. In [41], Thiémond gave an integer linear programming formulation for the problem and used techniques as cutting plane generation and branch and bound to tackle it (cf. also [20]). Quite recently, Shah proposed a genetic algorithm to calculate lower bounds for the star discrepancy [36].

In this paper we present a new randomized algorithm to approximate the star discrepancy. As the algorithm of Winker and Fang, it is based on threshold accepting, but with more problem specific knowledge added.

The paper is organized as follows. In Section 2 we describe the algorithm of Winker and Fang. In Section 3 we present a first version of our algorithm. The most important difference to the algorithm of Winker and Fang is a new non-uniform sampling strategy that takes into account the influence of the dimension d and topological characteristics of the given point set. In Section 4 we introduce the concept of critical test boxes, which are the boxes that lead to the largest discrepancy values, including the maximum value. We present rounding procedures which transform given test boxes into critical test boxes. With the help of these procedures and some other modifications, our algorithm achieves even better results, although at the cost of larger running times (see Table 6.1). In Section 5 we analyze the new sampling strategy and the rounding procedures in more depth. We provide comprehensive numerical tests in Section 6. The results indicate that our new algorithm is superior to all other known methods, especially in higher dimensions. The appendix contains some technical results necessary for our theoretical analysis in Section 5.

2. The Algorithm of Winker and Fang.

2.1. Notation. In addition to the notation introduced above, we make use of the following conventions.

For all positive integers $m \in \mathbb{N}$ we put $[m] := \{1, \dots, m\}$. If $r \in \mathbb{R}$, let $\lfloor r \rfloor := \max\{n \in \mathbb{Z} \mid n \leq r\}$. For the purpose of readability we sometime omit the $\lfloor \cdot \rfloor$ sign, i.e., whenever we write r where an integer is required, we implicitly mean $\lfloor r \rfloor$. For general $x, y \in [0, 1]^d$ we write $x \leq y$ if $x_j \leq y_j$ for all $j \in [d]$ and, equivalently, $x < y$ if $x_j < y_j$ for all $j \in [d]$.

For a given sequence $X = (x^i)_{i=1}^n$ in the d -dimensional unit cube $[0, 1]^d$, we define, next to the local discrepancy δ , the following functions. For all $y \in [0, 1]^d$ we set

$$\bar{\delta}(y) = \bar{\delta}(y, X) := \bar{A}(y, X) - V_y \quad \text{and} \quad \delta^*(y) = \delta^*(y, X) := \max\{\delta(y), \bar{\delta}(y)\}.$$

Then $d_\infty^*(X) = \max_{y \in \bar{\Gamma}(X)} \delta^*(y)$ as discussed in the introduction.

2.2. The algorithm of Winker and Fang. Threshold accepting is an integer optimization heuristic introduced by Dueck and Scheuer in [13]. Althöfer and Koschnik [1] showed that for suitably chosen parameters, threshold accepting converges to a global optimum if the number I of iterations tends to infinity. Winker and Fang [43] applied threshold accepting to compute the star discrepancy of a given n -point configuration. In the following, we give a short presentation of their algorithm. A flow diagram of the algorithm can be found in [43].

Initialization: The heuristic starts with choosing uniformly at random a starting point $x^c \in \bar{\Gamma}(X)$ and calculating $\delta^*(x^c) = \max\{\delta(x^c), \bar{\delta}(x^c)\}$. Note that throughout the description of the algorithm, x^c denotes the *currently* used search point.

Optimization: A number I of iterations is performed. In the t -th iteration, the algorithm chooses a point x^{nb} uniformly at random from a given *neighborhood*

$\mathcal{N}(x^c)$ of x^c and calculates $\delta^*(x^{nb})$. It then computes $\Delta\delta^* := \delta^*(x^{nb}) - \delta^*(x^c)$. If $\Delta\delta^* \geq T$ for a given (non-positive) threshold value T , then x^c is updated, i.e, the algorithm sets $x^c := x^{nb}$. With the help of the non-positive threshold it shall be avoided to get stuck in a bad local maximum x^c of δ^* —“local” with respect to the underlying neighborhood definition. The threshold value T changes during the run of the algorithm and ends up at zero. This should enforce the algorithm to end up at a local maximum of δ^* which is reasonably close to $d_\infty^*(X)$.

Neighborhood Structure: Let $x \in \bar{\Gamma}(X)$ be given. Let $\ell < n/2$ be an integer and put $k := 2\ell + 1$. We allow only a certain number of coordinates to change by fixing a value $mc \in [d]$ and choosing mc coordinates $j_1, \dots, j_{mc} \in [d]$ uniformly at random. For $j \in \{j_1, \dots, j_{mc}\}$ we consider the set of grid coordinates

$$\mathcal{N}_{k,j}(x) := \left\{ \gamma \in \bar{\Gamma}_j(X) \mid \max\{1, \phi_j^{-1}(x_j) - \ell\} \leq \phi_j^{-1}(\gamma) \leq \min\{|\bar{\Gamma}_j(X)|, \phi_j^{-1}(x_j) + \ell\} \right\},$$

where $\phi_j : |\bar{\Gamma}_j(X)| \rightarrow \bar{\Gamma}_j(X)$ is the ordering of the set $\bar{\Gamma}_j(X)$, i.e., $\phi_j(r) < \phi_j(s)$ for $r < s$. The *neighborhood* $\mathcal{N}_k^{j_1, \dots, j_{mc}}(x)$ of x of order k is the Cartesian product

$$\mathcal{N}_k^{j_1, \dots, j_{mc}}(x) := \hat{\mathcal{N}}_{k,1}(x) \times \dots \times \hat{\mathcal{N}}_{k,d}(x), \quad (2.1)$$

where $\hat{\mathcal{N}}_{k,j}(x) = \mathcal{N}_{k,j}(x)$ for $j \in \{j_1, \dots, j_{mc}\}$ and $\hat{\mathcal{N}}_{k,j}(x) = \{x_j\}$ otherwise. Clearly, $|\mathcal{N}_k^{j_1, \dots, j_{mc}}(x)| \leq (2\ell + 1)^{mc}$. We abbreviate $\mathcal{N}_k^{mc}(x) := \mathcal{N}_k^{j_1, \dots, j_{mc}}(x)$ if j_1, \dots, j_{mc} are mc coordinates chosen uniformly at random.

Threshold Values: The following procedure is executed prior to the algorithm itself. Let I be the total number of iterations to be performed by the algorithm and let $k \leq n$ and $mc \leq d$ be fixed. For each $t \in [\sqrt{I}]$, the procedure computes a pair (y^t, \tilde{y}^t) , where $y^t \in \bar{\Gamma}(X)$ is chosen uniformly at random and $\tilde{y}^t \in \mathcal{N}_k^{mc}(y^t)$, again chosen uniformly at random. It then calculates the values $T(t) := -|\delta^*(y^t) - \delta^*(\tilde{y}^t)|$. When all values $T(t)$, $t = 1, \dots, \sqrt{I}$, have been computed, the algorithm sorts them in increasing order. For a given $\alpha \in (0.9, 1]$, the $\alpha\sqrt{I}$ values closest to zero are selected as threshold sequence. The number J of iterations performed for each threshold value is $J = \alpha^{-1}\sqrt{I}$.

3. A First Improved Algorithm – TA_basic. Our first algorithm, `TA_basic`, builds on the algorithm of Winker and Fang as presented in the previous section. A preliminary, slightly different version of `TA_basic` can be found in [44]. This version was used in [11] to provide lower bounds for the comparison of the star discrepancies of different point sequences. In particular in higher dimensions it performed better than any other method tested by the authors.

Recall that the algorithm of Winker and Fang employs a uniform probability distribution on $\bar{\Gamma}(X)$ and the neighborhoods $\mathcal{N}_k^{mc}(x)$ for all random decisions.

Firstly, this is not appropriate for higher-dimensional inputs: In any dimension d it is most likely that the discrepancy of a set X is caused by test boxes with volume at least c , c some constant in $(0, 1)$. Thus in higher dimension d we expect the upper right corners of test boxes with large local discrepancy to have coordinates at least $c^{1/d}$. Thus it seems appropriate for higher dimensional sets X to increase the weight of those points in the grid $\bar{\Gamma}(X)$ with larger coordinates whereas we decrease the weight of the points with small coordinates.

Secondly, a uniform probability distribution does not take into account the topological characteristics of the point set X as, e.g., distances between the points in the grid $\bar{\Gamma}(X)$: If there is a grid cell $[x, y]$ in $\bar{\Gamma}(X)$ (i.e., $x, y \in \bar{\Gamma}(X)$) and

$\phi_j^{-1}(y_j) = \phi_j^{-1}(x_j) + 1$ for all $j \in [d]$, where ϕ_j is again the ordering of the set $\bar{\Gamma}_j(X)$ with large volume, we would expect that $\bar{\delta}(x)$ or $\delta(y)$ are also rather large.

Thus, on the one hand, it seem better to consider a modified probability measure on $\bar{\Gamma}(X)$ which accounts for the influence of the dimension and the topological characteristics of X . On the other hand, if n and d are large, we clearly cannot afford an elaborate precomputation of the modified probability weights.

To cope with this, the non-uniform sampling strategy employed by `TA_basic` consists of two steps:

- A continuous sampling step, where we select a point in the whole d -dimensional unit cube (or in a “continuous” neighborhood of x^c) with respect to a non-uniform (continuous) probability measure π^d , which is more concentrated in points with larger coordinates.
- A rounding step, where we round the selected point to the grid $\bar{\Gamma}(X)$.

In this way we address both the influence of the dimension and the topological characteristics of the point set X . This works without performing any precomputation of probability weights on $\bar{\Gamma}(X)$ – instead, the random generator, the change of measure on $[0, 1]^d$ from the d -dimensional Lebesgue measure to π^d , and our rounding procedure do this implicitly! Theoretical and experimental justifications for our non-uniform sampling strategy can be found in Section 5 and Section 6.

3.1. Sampling of Neighbors. In the following, we present how we modify the probability distribution over the neighborhood sets. Our non-uniform sampling strategy consists of the following two steps.

Continuous Sampling. Consider a point $x \in \bar{\Gamma}(X)$. For fixed $mc \in [d]$ let $j_1, \dots, j_{mc} \in [d]$ be pairwise different coordinates. For $j \in \{j_1, \dots, j_{mc}\}$ let $\varphi_j : [|\bar{\Gamma}_j(X) \cup \{0\}|] \rightarrow \bar{\Gamma}_j(X) \cup \{0\}$ be the ordering of the set $\bar{\Gamma}_j(X) \cup \{0\}$ (in particular $\varphi_j(1) = 0$). Let us now consider the real interval $C_{k,j}(x) := [\xi(x_j), \eta(x_j)]$ with

$$\xi(x_j) := \varphi(\max\{1, \varphi^{-1}(x_j) - \ell\}) \text{ and } \eta(x_j) := \varphi(\min\{|\bar{\Gamma}_j(X) \cup \{0\}|, \varphi^{-1}(x_j) + \ell\}).$$

Our new *neighborhood* $C_k^{j_1, \dots, j_{mc}}(x)$ of x of order k is the Cartesian product

$$C_k^{j_1, \dots, j_{mc}}(x) := \hat{C}_{k,1}(x) \times \dots \times \hat{C}_{k,d}(x), \quad (3.1)$$

where $\hat{C}_{k,j}(x) = C_{k,j}(x)$ for $j \in \{j_1, \dots, j_{mc}\}$ and $\hat{C}_{k,j}(x) = \{x_j\}$ otherwise. We abbreviate $C_k^{mc}(x) := C_k^{j_1, \dots, j_{mc}}(x)$ if j_1, \dots, j_{mc} are mc coordinates chosen uniformly at random.

Instead of endowing $C_k^{j_1, \dots, j_{mc}}(x)$ with the Lebesgue measure on the non-trivial components, we choose a different probability distribution which we describe in the following. First, let us consider the polynomial product measure

$$\pi^d(dx) = \otimes_{j=1}^d f(x_j) \lambda(dx_j) \text{ with density function } f : [0, 1] \rightarrow \mathbb{R}, r \mapsto dr^{d-1}$$

on $[0, 1]^d$; here $\lambda = \lambda^1$ should denote the one-dimensional Lebesgue measure. Notice that in dimension $d = 1$ we have $\pi^1 = \lambda$. Picking a random point $y \in [0, 1]^d$ with respect to the new probability measure π^d can easily be done in practice by sampling a point $z \in [0, 1]^d$ with respect to λ^d and then putting $y := (z_1^{1/d}, \dots, z_d^{1/d})$.

We endow $C_k^{j_1, \dots, j_{mc}}(x)$ with the probability distribution induced by the polynomial product measure on the mc non-trivial components $C_{k,j_1}(x), \dots, C_{k,j_{mc}}(x)$. To be more explicit, we map each $C_{k,j}(x)$, $j \in \{j_1, \dots, j_{mc}\}$, to the unit interval $[0, 1]$ by

$$\Psi_j : C_{k,j}(x) \rightarrow [0, 1], r \mapsto \frac{r^d - (\xi(x_j))^d}{(\eta(x_j))^d - (\xi(x_j))^d}.$$

Recall that $\xi(x_j) := \min C_{k,j}(x)$ and $\eta(x_j) := \max C_{k,j}(x)$. The inverse mapping Ψ_j^{-1} is then given by

$$\Psi_j^{-1} : [0, 1] \rightarrow C_{k,j}, s \mapsto \left(((\eta(x_j))^d - (\xi(x_j))^d)s + (\xi(x_j))^d \right)^{1/d}.$$

If we want to sample a random point $y \in C_k^{j_1, \dots, j_{mc}}(x)$, we randomly choose scalars s_1, \dots, s_{mc} in $[0, 1]$ with respect to λ and put $y_{j_i} := \Psi_{j_i}^{-1}(s_i)$ for $i = 1, \dots, mc$. For indices $j \notin \{j_1, \dots, j_{mc}\}$ we set $y_j := x_j$.

Rounding Procedure. We round the point y once up and once down to the nearest points y^+ and y^- in $\bar{\Gamma}(X)$. More precisely, for all $j \in [d]$, let $y_j^+ := \min\{x_j^i \in \bar{\Gamma}_j(X) \mid y_j \leq x_j^i\}$. If $y_j \geq \min \bar{\Gamma}_j(X)$ we set $y_j^- := \max\{x_j^i \in \bar{\Gamma}_j(X) \mid y_j \geq x_j^i\}$ and in case $y_j < \min \bar{\Gamma}_j(X)$, we set $y_j^- := \max \Gamma_j(X)$.

Obviously, $A(y^+, X) = A(y, X)$ and thus, $\delta(y^+) = V_{y^+} - A(y^+, X) \geq V_y - A(y, X) = \delta(y)$. Similarly, if $y_j \geq \min \Gamma_j(X)$ for all $j \in [d]$, we have $\bar{A}(y^-, X) = \bar{A}(y, X)$. Hence, $\bar{\delta}(y^-) = \bar{A}(y^-, X) - V_{y^-} \geq \bar{A}(y, X) - V_y = \bar{\delta}(y)$. If $y_j < \min \Gamma_j(X)$ for at least one $j \in [d]$ we have $\bar{\delta}(y) \leq 0$ since $\bar{A}(y, X) = 0$. But we also have $A(y, X) = 0$ and thus $\delta^*(y) = \delta(y) \leq \delta(y^+)$. Putting everything together, we have shown that $\max\{\delta(y^+), \bar{\delta}(y^-)\} \geq \delta^*(y)$.

Since it is only of insignificant additional computational cost to also compute $\bar{\delta}(y^{-,-})$ where $y_j^{-,-} := y_j^-$ for all $j \in [d]$ with $y_j \geq \min \bar{\Gamma}_j(X)$ and $y_j^{-,-} := \min \bar{\Gamma}_j(X)$ for j with $y_j < \min \bar{\Gamma}_j(X)$, we also do that in case at least one such j with $y_j < \min \bar{\Gamma}_j(X)$ exists.

For sampling a neighbor x^{nb} of x^c the algorithm thus does the following. First, it samples mc coordinates $j_1, \dots, j_{mc} \in [d]$ uniformly at random. Then it samples a point $y \in C_k^{j_1, \dots, j_{mc}}(x^c)$ as described above, computes the rounded grid points y^+ , y^- , and $y^{-,-}$ and computes the discrepancy $\delta_\Gamma^*(y) := \max\{\delta(y^+), \bar{\delta}(y^-), \bar{\delta}(y^{-,-})\}$ of the rounded grid points. The subscript Γ shall indicate that we consider the rounded grid points. As in the algorithm of Winker and Fang, `TA_basic` updates x^c if and only if $\Delta\delta^* = \delta_\Gamma^*(y) - \delta^*(x^c) \geq T$, where T denotes the current threshold. In this case we always update x^c with the best rounded test point, i.e., we update $x^c := y^+$ if $\delta_\Gamma^*(y) = \delta(y^+)$, $x^c := y^-$ if $\delta_\Gamma^*(y) = \bar{\delta}(y^-)$, and $x^c := y^{-,-}$ otherwise.

3.2. Sampling of the Starting Point. Similar to the probability distribution on the neighborhood sets, we sample the starting point x^c as follows. First, we sample a point x from $[0, 1]^d$ according to π^d . We then round x up and down to x^+ , x^- , and $x^{-,-}$, respectively and again we set $x^c := x^+$ if $\delta_\Gamma^*(x) = \delta(x^+)$, $x^c := x^-$ if $\delta_\Gamma^*(x) = \bar{\delta}(x^-)$, and $x^c := x^{-,-}$ otherwise.

3.3. Computation of Threshold Sequence. The modified neighborhood sampling is also used for computing the sequence of threshold values. If we want the algorithm to perform I iterations, we compute the threshold sequence as follows. For each $t \in [\sqrt{I}]$ we sample a pair (y^t, \tilde{y}^t) , where $y^t \in \bar{\Gamma}(X)$ is sampled as is the starting point and $\tilde{y}^t \in \bar{\Gamma}(X)$ is a neighbor of y^t , sampled according to the procedure described in Section 3.1. The thresholds $|\delta^*(y^t) - \delta^*(\tilde{y}^t)|$ are sorted in increasing order and each threshold will be used for \sqrt{I} iterations of `TA_basic`. Note that by this choice, we are implicitly setting $\alpha := 1$ in the notion of the algorithm of Winker and Fang.

4. Further Improvements – Algorithm `TA_improved`. In the following, we present further modifications which we applied to the basic algorithm `TA_basic`. We call the new, enhanced algorithm `TA_improved`.

The main improvements, which we describe in more detail below, are **(i)** a further reduction of the search space by introducing new rounding procedures (“snapping”), **(ii)** shrinking neighborhoods and growing number of search directions, and **(iii)** separate optimization of δ and $\bar{\delta}$.

4.1. Further Reduction of the Search Space. We mentioned that for calculating the star discrepancy it is sufficient to test just the points $y \in \bar{\Gamma}(X)$ and to calculate $\delta^*(y)$, cf. equation (1.1). Therefore $\bar{\Gamma}(X)$ has been the search space we have considered so far. But it is possible to reduce the cardinality of the search space even further.

We obtain the reduction of the search space via a rounding procedure which we call *snapping*. We now discuss the underlying concept of critical points (or test boxes), which is an important element in the modified algorithm. For $y \in [0, 1]^d$ we define

$$S_j(y) := \prod_{i=1}^{j-1} [0, y_i] \times \{y_j\} \times \prod_{k=j+1}^d [0, y_k], \quad j = 1, \dots, d.$$

We say that $S_j(y)$ is a $\delta(X)$ -critical surface if $S_j(y) \cap \{x^1, \dots, x^n\} \neq \emptyset$ or $y_j = 1$. We call y a $\delta(X)$ -critical point if for all $j \in [d]$ the surfaces $S_j(y)$ are $\delta(X)$ -critical. Let \mathcal{C} denote the set of $\delta(X)$ -critical points in $[0, 1]^d$.

Further, let $\bar{S}_j(y)$ be the closure of $S_j(y)$, i.e.,

$$\bar{S}_j(y) := \prod_{i=1}^{j-1} [0, y_i] \times \{y_j\} \times \prod_{k=j+1}^d [0, y_k], \quad j = 1, \dots, d.$$

We say $\bar{S}_j(y)$ is a $\bar{\delta}(X)$ -critical surface if $\bar{S}_j(y) \cap \{x_1, \dots, x_n\} \neq \emptyset$. If for all $j \in [d]$ the surfaces $\bar{S}_j(y)$ are $\bar{\delta}(X)$ -critical, then we call y a $\bar{\delta}(X)$ -critical point. Let $\bar{\mathcal{C}}$ denote the set of $\bar{\delta}(X)$ -critical points in $[0, 1]^d$. We call y a $\delta^*(X)$ -critical point if $y \in \mathcal{C}^* := \mathcal{C} \cup \bar{\mathcal{C}}$.

For $j \in [d]$ let $\nu_j := |\bar{\Gamma}_j(X)|$, and let again $\phi_j : [\nu_j] \rightarrow \bar{\Gamma}_j(X)$ denote the ordering of $\bar{\Gamma}_j(X)$. Let $\Phi : [\nu_1] \times \dots \times [\nu_d] \rightarrow \bar{\Gamma}(X)$ be the mapping with components ϕ_j , $j = 1, \dots, d$. We say that a multi-index $(i_1, \dots, i_d) \in [\nu_1] \times \dots \times [\nu_d]$ is a $\delta(X)$ -critical multi-index if $\Phi(i_1, \dots, i_d)$ is a $\delta(X)$ -critical point. We use similar definitions in cases where we deal with $\bar{\delta}(X)$ or $\delta^*(X)$.

LEMMA 4.1. *Let $X = (x^i)_{i=1}^n$ be a sequence in $[0, 1]^d$. Let $\mathcal{C} = \mathcal{C}(X)$, $\bar{\mathcal{C}} = \bar{\mathcal{C}}(X)$, and $\mathcal{C}^* = \mathcal{C}^*(X)$ be as defined above. Then \mathcal{C} , $\bar{\mathcal{C}}$ and \mathcal{C}^* are non-empty subsets of $\bar{\Gamma}(X)$. Furthermore,*

$$\sup_{y \in [0, 1]^d} \delta(y) = \max_{y \in \mathcal{C}} \delta(y), \quad \sup_{y \in [0, 1]^d} \bar{\delta}(y) = \max_{y \in \bar{\mathcal{C}}} \bar{\delta}(y) \quad \text{and} \quad \sup_{y \in [0, 1]^d} \delta^*(y) = \max_{y \in \mathcal{C}^*} \delta^*(y).$$

Proof. The set \mathcal{C} is not empty, since it contains the point $(1, \dots, 1)$. Let $y \in \mathcal{C}$. By definition, we find for all $j \in [d]$ an index $\sigma(j) \in [n]$ with $y_j = x_j^{\sigma(j)}$ or we have $y_j = 1$. Therefore $y \in \bar{\Gamma}(X)$. Let $z \in [0, 1]^d \setminus \mathcal{C}$. Since $\delta(z) = 0$ if $z_j = 0$ for any index j , we may assume $z_j > 0$ for all j . As $z \notin \mathcal{C}$ there exists a $j \in [d]$ where $S_j(z)$ is not $\delta(X)$ -critical. In particular, we have $z_j < 1$. Let now $\tau \in \bar{\Gamma}_j(X)$ be the smallest value with $z_j < \tau$. Then the point $\hat{z} := (z_1, \dots, z_{j-1}, \tau, z_{j+1}, \dots, z_d)$ fulfills $V_{\hat{z}} > V_z$. Furthermore, the sets $[0, \hat{z}] \setminus [0, z]$ and X are disjoint. So $[0, \hat{z}]$ and $[0, z]$ contain the same points of X . In particular we have $A(\hat{z}, X) = A(z, X)$ and thus, $\delta(\hat{z}) > \delta(z)$. This argument verifies $\sup_{y \in [0, 1]^d} \delta(y) = \max_{y \in \mathcal{C}} \delta(y)$. The remaining statements of Lemma 4.1 can be proved with similar simple arguments. \square

We now describe how to use this concept in our algorithm. Let us first explain how we sample a neighbor x^{nb} of a given point x^c . The procedure starts exactly as described in Section 3.1. That is, we first sample mc coordinates $j_1, \dots, j_{mc} \in [d]$ uniformly at random. Next, we sample $y \in C_k^{j_1, \dots, j_{mc}}(x^c)$ according to the probability distribution induced by the polynomial product measure π^d on the non-trivial components of $C_k^{j_1, \dots, j_{mc}}(x^c)$, cf. Section 3.1. Again we round y up and down to the nearest grid points y^+ , y^- and $y^{-,-}$, respectively. We then apply the following snapping procedures¹.

Snapping down. We aim at finding a $\bar{\delta}(X)$ -critical point $y^{-,sn} \leq y^-$ such that the closed box $[0, y_j^{-,sn}]_{j=1}^d$ contains exactly the same points of X as the box $[0, y_j^-]_{j=1}^d$. We achieve this by simply setting for all $j \in [d]$

$$y_j^{-,sn} := \max\{x_j^i \mid i \in [n], x^i \in [0, y^-]\}.$$

From the algorithmic perspective, we initialize $y^{-,sn} := (0, \dots, 0)$ and check for each index $i \in [n]$ whether $x^i \in [0, y^-]$. If so, we check for all $j \in [d]$ whether $x_j^i \leq y_j^{-,sn}$ and update $y_j^{-,sn} := x_j^i$ otherwise.

Snapping up². Whereas snapping down was an easy task to do, the same is not true for *snapping up*, i.e., rounding a point to a $\delta(X)$ -critical one. More precisely, given a point y^+ , there are in general multiple $\delta(X)$ -critical points $y^{+,sn} \geq y^+$ such that the open box created by $y^{+,sn}$ contains only those points which are also contained in $[0, y^+)$.

Given that we want to perform only one snapping up procedure per iteration, we use the following random version of snapping upwards. In the beginning, we initialize $y^{+,sn} := (1, \dots, 1)$. Furthermore, we pick a permutation σ of $[d]$ uniformly at random from the set S_d of all permutations of set $[d]$. For each point $x \in \{x^i \mid i \in [n]\}$ we now do the following. If $x \in [0, y^+)$ or $x_j \geq y_j^{+,sn}$ for at least one $j \in [d]$, we do nothing. Otherwise we update $y_{\sigma(j)}^{+,sn} := x_{\sigma(j)}$ for the smallest $j \in [d]$ with $x_{\sigma(j)} \geq y_{\sigma(j)}^{+,sn}$. After this update, x is no longer inside the half-open box generated by $y^{+,sn}$.

Note that snapping up is subject to randomness as the $\delta(X)$ -critical point obtained by our snapping procedure can be different for different permutations $\sigma \in S_d$.

The complexity of both snapping procedures is of order $O(nd)$. In our experiments, the snapping procedures caused a delay in the (wall clock) running time by a factor of approximately two, if compared to the running time of `TA_basic`. It is not difficult to verify the following.

LEMMA 4.2. *Let X be a given n -point sequence in $[0, 1]^d$. For all $y \in [0, 1]^d$, the point $y^{+,sn}$, computed as described above, is $\delta(X)$ -critical and both $y^{-,sn}$ and $y^{-,-,sn}$ are $\bar{\delta}(X)$ -critical.*

In the run of the algorithm we now do the following. Given that we start in some grid point x^c , we sample $y \in C_k^{mc}(x^c)$ and we round y to the closest grid points $y^+, y^-, y^{-,-} \in \bar{\Gamma}(X)$ as described in Section 3.1. Next we compute the $\delta(X)$ -critical point $y^{+,sn}$, the $\bar{\delta}(X)$ -critical point $y^{-,sn}$, and, if $y^- \neq y^{-,-}$, we also compute the $\bar{\delta}(X)$ -critical point $y^{-,-,sn}$. We decide to update x^c if $\Delta\delta^* = \delta^{*,sn}(y) - \delta^{*,sn}(x^c) \geq T$, where T is the current threshold, $\delta^{*,sn}(y) := \max\{\delta(y^{+,sn}), \bar{\delta}(y^{-,sn}), \bar{\delta}(y^{-,-,sn})\}$, and $\delta^{*,sn}(x^c)$ is the value as was computed in the iteration where x^c was updated last. Note that we do not update x^c with any of the critical points $y^{+,sn}$, $y^{-,sn}$, or $y^{-,-,sn}$

¹The snapping procedure is the same for y^- and $y^{-,-}$. Therefore, we describe it for y^- only.

²While this is a slight misnomer, we still use this phrase to ease readability in what follows.

but only replace x^c with the simple rounded grid points y^+ , y^- , or $y^{-,-}$, respectively. More precisely, we update $x^c := y^+$ if $\delta^{*,sn}(y) = \delta(y^{+,sn})$, $x^c := y^-$ if $\delta^{*,sn}(y) = \bar{\delta}(y^{-,sn})$, and $x^c := y^{-,-}$, otherwise.

4.1.1. Computation of the Starting Point and the Threshold Sequence.

When computing the starting point x^c we first sample a random point x from $[0, 1]^d$ according to π^d (see Section 3.1) and compute x^+ and x^- , and, if applicable, $x^{-,-}$. We also compute the $\delta(X)$ - and $\bar{\delta}(X)$ -critical points $x^{+,sn}$, $x^{-,sn}$, and $x^{-,-,sn}$ and set $\delta^{*,sn}(x) := \max\{\delta(x^{+,sn}), \bar{\delta}(x^{-,sn}), \bar{\delta}(x^{-,-,sn})\}$. We put $x^c := x^+$ if $\delta^{*,sn}(x) = \delta(x^{+,sn})$, $x^c := x^-$ if $\delta^{*,sn}(x) = \bar{\delta}(x^{-,sn})$, and $x^c := x^{-,-}$, otherwise.

For computing the threshold sequence, we also use the $\delta(X)$ - and $\bar{\delta}(X)$ -critical $\delta^{*,sn}$ -values. That is, for $t = 1, \dots, \sqrt{I}$ we compute t -th pair (y^t, \tilde{y}^t) by first sampling a random starting point y^t as described above (i.e., $y^t \in \{x^+, x^-, x^{-,-}\}$ for some x sampled from $[0, 1]^d$ according to π^d and $y^t = x^+$ if $\delta^{*,sn}(x) = \delta(x^{+,sn})$, $y^t = x^-$ if $\delta^{*,sn}(x) = \bar{\delta}(x^{-,sn})$, and $y^t = x^{-,-}$ otherwise). We then compute a neighbor $\tilde{y}^t \in C_k^{mc}(y^t)$ and the maximum of the discrepancy of the $\delta(X)$ - and $\bar{\delta}(X)$ -critical points $\delta^{*,sn}(\tilde{y}^t) := \max\{\delta(\tilde{y}^{t,+}, sn), \bar{\delta}(\tilde{y}^{t,-}, sn), \bar{\delta}(\tilde{y}^{t,-,-}, sn)\}$. Finally, we sort the threshold values $T(t) := -|\delta^{*,sn}(y^t) - \delta^{*,sn}(\tilde{y}^t)|$, $t = 1, \dots, \sqrt{I}$ in increasing order. This will be our threshold sequence.

4.2. Shrinking Neighborhoods and Growing Number of Search Directions. We add the concept of shrinking neighborhoods, i.e., we consider neighborhoods that decrease in size during the run of the algorithm. The intuition here is the following. In the beginning, we want the algorithm to make large jumps. This allows it to explore different regions of the search space. However, towards the end of the algorithm we want it to become more local, allowing it to explore large parts of the local neighborhood. We implement this idea by iteratively shrinking the k -value. At the same time, we increase the mc -value, letting the algorithm explore the local neighborhood more thoroughly.

More precisely, we do the following. In the beginning we set $\ell := (n - 1)/2$ and $mc := 2$. That is, the algorithm is only allowed to change few coordinates of the current search point but at the same time it can make large jumps in these directions. Recall that $k = 2\ell + 1$. In the t -th iteration (out of a total number of I iterations) we then update

$$\ell := \frac{n-1}{2} \cdot \frac{I-t}{I} + \frac{t}{I} \quad \text{and} \quad mc := 2 + \frac{t}{I}(d-2).$$

For the computation of the threshold sequence, we equivalently scale k and mc by initializing $\ell := (n - 1)/2$ and $mc := 2$ and then setting for the computation of the t -th pair (y^t, \tilde{y}^t)

$$\ell := \frac{n-1}{2} \cdot \frac{\sqrt{I}-t}{\sqrt{I}} + \frac{t}{\sqrt{I}} \quad \text{and} \quad mc := 2 + \frac{t}{\sqrt{I}}(d-2).$$

Recall that we compute a total number of \sqrt{I} threshold values.

4.3. Separate Optimization of δ and $\bar{\delta}$. Our last modification is based on the intuition that if the star discrepancy obtained by an open, sub-proportionally filled box (i.e., there exists a $y \in \bar{\Gamma}(X)$ such that $d_\infty^*(X) = \delta(y)$), then one might assume that there are many points \tilde{y} with large $\delta(\tilde{y})$ -values. Conversely, if the discrepancy is obtained by a closed, over-proportionally filled box (i.e., there exists a $y \in \bar{\Gamma}(X)$ such

that $d_\infty^*(X) = \bar{\delta}(y)$, then there should be multiple points \tilde{y} with large $\bar{\delta}(\tilde{y})$ -values. This intuition triggered us to test the following split variant of the algorithm.

In the δ -version of the algorithm, we only consider open test boxes. That is, whenever we want to sample a random starting point [a random neighbor], we proceed exactly as described in Section 4.1 but instead of computing both y^+ and y^- (and, potentially $y^{-,-}$) as well as the $\delta(X)$ - and $\bar{\delta}(X)$ -critical points $y^{+,sn}$, $y^{-,sn}$, and $y^{-,-,sn}$ in the notation of Section 4.1, we only compute y^+ [and $y^{+,sn}$], and we initialize $x^c := y^+$ [we update $x^c := y^+$ if and only if $\Delta\delta = \delta(y^{+,sn}) - \delta((x^c)^{+,sn}) \geq T$, where T again denotes the current threshold].

The $\bar{\delta}$ -version is symmetric. We compute both y^- and $y^{-,-}$ as well as the $\bar{\delta}(X)$ -critical points $y^{-,sn}$ and $y^{-,-,sn}$, and we initialize $x^c := y^-$ or $x^c := y^{-,-}$ [we update $x^c := y^-$ or $x^c := y^{-,-}$ if and only if $\Delta\bar{\delta} = \max\{\bar{\delta}(y^{-,sn}), \bar{\delta}(y^{-,-,sn})\} - \bar{\delta}((x^c)^{-,sn}) \geq T$]. Note that only δ -values (or $\bar{\delta}$ -values, respectively) are considered for the computation of the threshold sequence as well.

The algorithm is now the following. We perform I iterations of the δ -version of the algorithm and I iterations of the $\bar{\delta}$ -version. We then output the maximum value obtained in either one of the two versions.

It should be noted that a large proportion of the computational cost of `TA_improved` lies in the snapping procedures. Thus, running I iterations of the δ -version followed by I iterations of the $\bar{\delta}$ -version has a comparable running time to running I iterations of an algorithm of the “mixed” form where we snap each point up and down to the $\delta(X)$ - and $\bar{\delta}(X)$ -critical grid points. Furthermore, as most modern CPUs are multi-core and able to run several programs in parallel, the actual wall-clock cost of switching from `TA_basic` to the split version of `TA_improved` may be smaller still.

5. Theoretical Analysis. From our main innovations, namely the non-uniform sampling strategy and the rounding procedures “snapping up” and “snapping down”, we already analyzed the snapping procedures and proved that they enlarge the quality of our estimates. The analysis of the non-uniform sampling strategy is much more complicated. One reason is that our sampling strategy strongly interacts with the search heuristic threshold accepting. That is why we confine ourselves to the analysis of the pure non-uniform sampling strategy without considering threshold accepting.

In Section 5.1 we prove that sampling in the d -dimensional unit cube with respect to the probability measure π^d instead of λ^d leads to larger values of the local discrepancy. In Section 5.2 we verify that for $d = 1$ sampling with respect to the probability distribution induced on $\bar{\Gamma}(X)$ by sampling with respect to π^d in $[0, 1]^d$ and then rounding to the grid $\bar{\Gamma}(X)$ leads to better discrepancy estimates than the uniform distribution on $\bar{\Gamma}(X)$. We comment also on the case $d \geq 2$. In Section 5.3 we prove that for random point sets X the probability of $x \in \bar{\Gamma}(X)$ being a critical point is essentially an increasing function of the position of its coordinates x_j in the ordered sets $\bar{\Gamma}_j(X)$, $j = 1, \dots, d$. Recall that critical points yield higher values of the function δ^* and include the point that leads to its maximum value. Thus the analysis in Section 5.3 serves as another justification of choosing a probability measure on the neighborhoods which weights points with larger coordinates stronger than points with smaller coordinates.

5.1. Analysis of Random Sampling with Respect to λ^d and π^d . Here we want to show that sampling in the d -dimensional unit cube with respect to the non-uniform probability measure π^d leads to superior results than sampling with respect to the Lebesgue measure λ^d .

Before we start with the theoretical analysis, let us give a strong indication that our non-uniform sampling strategy is much more appropriate in higher dimension than a uniform sampling strategy. In [43] Winker and Fang chose in each of the dimensions $d = 4, 5, 6$ in a random manner 10 lattice points sets, cf. also our Section 6. They calculated the discrepancy of these sets exactly. If η_d denotes the average value of the coordinates of the points y with $\delta^*(y) = \sup_{z \in [0,1]^d} \delta^*(z)$, we get $\eta_4 = 0.799743$, $\eta_5 = 0.840825$, and $\eta_6 = 0.873523$. The expected coordinate value μ_d of a point x , randomly sampled from $[0, 1]^d$ with respect to the measure π^d , is $\mu_d = d/(d + 1)$. So we get $\mu_4 = 0.8$, $\mu_5 = 0.8\bar{3}$, and $\mu_6 = 0.857143$. Note that for using λ^d instead of π^d the expected coordinate value is only 0.5 for all dimensions.

5.1.1. Random Sampling in the Unit Cube with Respect to λ^d . We analyze the setting, where we sample in $[0, 1]^d$ with respect to λ^d to maximize the objective function δ . A similar analysis for $\bar{\delta}$ is technically more involved than the proof of Proposition 5.1. Furthermore, it leads to a less clear and also worse result. We comment on this at the end of this subsection.

PROPOSITION 5.1. *Let $\varepsilon \in (0, 1)$, let $n, d \in \mathbb{N}$, and let $X = (x^i)_{i=1}^n$ be a sequence in $[0, 1]^d$. Let $x^* = x^*(X) \in [0, 1]^d$ satisfy $\delta(x^*) = \sup_{x \in [0, 1]^d} \delta(x)$. Let us assume that $V_{x^*} \geq \varepsilon$. Consider a random point $r \in [0, 1]^d$, sampled with respect to the probability measure λ^d . If $P_\varepsilon^\lambda = P_\varepsilon^\lambda(X)$ denotes the probability of the event $\{r \in [0, 1]^d \mid \delta(x^*) - \delta(r) \leq \varepsilon\}$, then*

$$P_\varepsilon^\lambda \geq \frac{1}{d!} \frac{\varepsilon^d}{V_{x^*}^{d-1}} \geq \frac{\varepsilon^d}{d!}. \quad (5.1)$$

This lower bound is sharp in the sense that there exist sequences of point configurations $\{X^{(k)}\}$ such that $\lim_{k \rightarrow \infty} d! \varepsilon^{-d} P_\varepsilon^\lambda(X^{(k)})$ converges to 1 as ε tends to zero.

Let additionally $\epsilon \in (0, 1)$ and $R \in \mathbb{N}$. Consider random points $r^1, \dots, r^R \in [0, 1]^d$, sampled independently with respect to λ^d , and put $\delta^R := \max_{i=1}^R \delta(r^i)$. If

$$R \geq |\ln(\epsilon)| \left| \ln \left(1 - \frac{\varepsilon^d}{d!} \right) \right|^{-1}, \quad (5.2)$$

then $\delta(x^) - \delta^R \leq \epsilon$ with probability at least $1 - \epsilon$.*

Notice, that the case $V_{x^*} < \varepsilon$ left out in Proposition 5.1 is less important for us, since our main goal is to find a good lower bound for the star-discrepancy $d_\infty^*(X)$. Indeed, the approximation of $d_\infty^*(X)$ up to an admissible error ε is a trivial task if $d_\infty^*(X) \leq \varepsilon$. If $d_\infty^*(X) > \varepsilon$, then $V_{x^*} < \varepsilon$ implies $\delta(x^*) < d_\infty^*(X)$, and the function $\bar{\delta}$ plays the significant role.

Proof. For $x \leq x^*$ we get

$$\delta(x) = V_x - \frac{1}{n} A(x, X) \geq \delta(x^*) - (V_{x^*} - V_x).$$

Therefore the Lebesgue measure of the set

$$A_\varepsilon(x^*) := \{x \in [0, 1]^d \mid x \leq x^*, V_{x^*} - V_x \leq \varepsilon\} \quad (5.3)$$

is a lower bound for P_ε^λ . Due to Proposition A.2, we have for $d \geq 2$

$$\lambda^d(A_\varepsilon(x^*)) = \frac{1}{d!} \frac{\varepsilon^d}{V_{x^*}^{d-1}} \sum_{k=0}^{\infty} b_k(d) \left(\frac{\varepsilon}{V_{x^*}} \right)^k,$$

with positive coefficients $b_k(d)$. In particular, we have $b_0(d) = 1$. Thus,

$$\lambda^d(A_\varepsilon(x^*)) \geq \frac{1}{d!} \frac{\varepsilon^d}{V_{x^*}^{d-1}} \geq \frac{\varepsilon^d}{d!}, \quad (5.4)$$

and this estimate is obviously also true for $d = 1$. Let us now consider for sufficiently large $k \in \mathbb{N}$ point configurations $X^{(k)} = (x^{(k),i})_{i=1}^n$, where

$$x_1^{(k),1} = \dots = x_1^{(k),n} = k/(k+1) > \varepsilon \quad (5.5)$$

and $x_j^{(k),i} < k/(k+1) - \varepsilon$ for all $i \in [n]$, $j > 1$. Then obviously $x^*(X^{(k)}) = (k/(k+1), 1, \dots, 1)$, and it is easy to see that $P_\varepsilon^\lambda(X^{(k)}) = \lambda^d(A_\varepsilon(x^*))$. From Proposition A.2 we get

$$\lambda^d(A_\varepsilon(x^*)) = \left(\frac{k+1}{k}\right)^{d-1} \frac{\varepsilon^d}{d!} \left(1 + O\left(\frac{k+1}{k} \varepsilon\right)\right).$$

This proves that estimate (5.1) is sharp. Notice that we assumed (5.5) only for simplicity. Since $\delta(x^*(X))$ is continuous in X , we can find for fixed k an open set of point configurations doing essentially the same job as $X^{(k)}$.

Assume now $\delta(x^*) - \delta^R > \varepsilon$, i.e., $\delta(x^*) - \delta(r^i) > \varepsilon$ for all $i \leq R$. The probability of this event is at maximum $(1 - \varepsilon^d/d!)^R$. This probability is bounded from above by ε if R satisfies (5.2). \square

For $d \geq 1$ and $\varepsilon \leq 1/2$ we have $|\ln(1 - \varepsilon^d/d!)|^{-1} \sim d! \varepsilon^{-d}$. In this case we can only assure that δ^R is an ε -approximation of $\sup_{x \in [0,1]^d} \delta(x)$ with a certain probability if the number R of randomly sampled points is super-exponential in d .

Let us end this section with some comments on the setting where we are only interested in maximizing $\bar{\delta}$. If for given $\varepsilon > 0$, $X \in [0,1]^{nd}$ the maximum of $\bar{\delta}$ is achieved in $\bar{x} = \bar{x}(X) \in [0,1]^d$, and if we want to know the probability of the event $\{r \in [0,1]^d \mid \bar{\delta}(\bar{x}) - \bar{\delta}(r) \leq \varepsilon\}$, there seems to be no alternative to estimating $\lambda^d(U(\bar{x}))$, where

$$U(\bar{x}) := \{r \in [0,1]^d \mid \bar{x} \leq r, V_r - V_{\bar{x}} \leq \varepsilon\}.$$

It is easy to see that $\lambda^d(U(\bar{x}(X)))$ approaches zero if one of the coordinates of \bar{x} tends to 1 – regardless of ε and $V_{\bar{x}}$. We omit a tedious error analysis to cover the $\bar{\delta}$ -setting.

5.1.2. Random Sampling in the Unit Cube with Respect to π^d . Similarly as in the preceding section, we analyze here the setting where, in order to maximize δ , we sample in $[0,1]^d$ with respect to π^d .

PROPOSITION 5.2. *Let $\varepsilon, d, n, X = (x^i)_{i=1}^n$ and x^* as in Proposition 5.1. Again assume $V_{x^*} \geq \varepsilon$. Consider a random point $r \in [0,1]^d$, sampled with respect to the probability measure π^d . If $P_\varepsilon^\pi = P_\varepsilon^\pi(X)$ denotes the probability of the event $\{r \in [0,1]^d \mid \delta(x^*) - \delta(r) \leq \varepsilon\}$, then $P_\varepsilon^\pi \geq \varepsilon^d$. This lower bound is sharp, since there exists a point configuration X such that $P_\varepsilon^\pi(X) = \varepsilon^d$.*

Let additionally $\epsilon \in (0,1)$ and $R \in \mathbb{N}$. Consider random points $r^1, \dots, r^R \in [0,1]^d$, sampled independently with respect to π^d , and put $\delta^R := \max_{i=1}^R \delta(r^i)$. If

$$R \geq |\ln(\epsilon)| |\ln(1 - \varepsilon^d)|^{-1}, \quad (5.6)$$

then $\delta(x^) - \delta^R \leq \epsilon$ with probability at least $1 - \epsilon$.*

Proof. Clearly $P_\varepsilon^\pi \geq \pi^d(A_\varepsilon(x^*))$, where $A_\varepsilon(x^*)$ is defined as in (5.3). Due to Proposition A.4 we have $\pi^d(A_\varepsilon(x^*)) \geq \varepsilon^d$. Let us now consider the point configuration X , where $x_1^1 = \dots = x_1^n = \varepsilon$ and $x_j^i < \varepsilon$ for all $i \in [n]$, $j > 1$. Furthermore, at least an ε^{-1} -fraction of the points should be equal to $(\varepsilon, 0, \dots, 0)$. Then obviously $x^*(X) = (\varepsilon, 1, \dots, 1)$ and $P_\varepsilon^\lambda(X) = \pi^d(A_\varepsilon(x^*)) = \varepsilon^d$.

Let us now assume that $\delta(x^*) - \delta^R > \varepsilon$, i.e., $\delta(x^*) - \delta(r_i) > \varepsilon$ for all $i \leq R$. This happens with probability not larger than $(1 - \varepsilon^d)^R$. Therefore we have $(1 - \varepsilon^d)^R \leq \varepsilon$ if R satisfies (5.6). \square

If $d \geq 1$ and $\varepsilon \leq 1/2$, then $|\ln(1 - \varepsilon^d)|^{-1} \sim \varepsilon^{-d}$. Here the number of iterations R ensuring with a certain probability that δ^R is an ε -approximation of $\sup\{\delta(x) \mid x \in [0, 1]^d\}$ is still exponential in d , but at least not super-exponential as in the previous section.

Altogether we see that a simple sampling algorithm relying on the probability measure π^d rather than on λ^d is more likely to find larger values of δ .

5.2. Analysis of Rounding to the Coordinate Grid. As described in Sections 3.1 and 3.2, our non-uniform sampling strategy on the grids $\bar{\Gamma}(X)$ and $\Gamma(X)$ for the objective functions δ and $\bar{\delta}$ consists of sampling in $[0, 1]^d$ with respect to π^d and then rounding the sampled point y up and down to grid points y^+ and y^- , respectively. This induces discrete probability distributions $w_u = (w_u(z))_{z \in \bar{\Gamma}(X)}$ and $w_l = (w_l(z))_{z \in \Gamma(X)}$ on $\bar{\Gamma}(X)$ and $\Gamma(X)$, respectively. If we use additionally the rounding procedures “snapping up” and “snapping down”, as described in Section 4.1, this will lead to modified probability distributions $w_u^{sn} = (w_u^{sn}(z))_{z \in \bar{\Gamma}(X)}$ and $w_l^{sn} = (w_l^{sn}(z))_{z \in \Gamma(X)}$ on $\bar{\Gamma}(X)$ and $\Gamma(X)$, respectively. In dimension $d = 1$ the probability distributions w_u and w_u^{sn} as well as w_l and w_l^{sn} are equal, since every test box is a critical one. Essentially we prove in the next section that in the one-dimensional case sampling with respect to the probability distributions $w_u = w_u^{sn}$ [$w_l = w_l^{sn}$] leads to larger values of δ [$\bar{\delta}$] than sampling with respect to the uniform distribution on $\bar{\Gamma}(X)$ [$\Gamma(X)$].

5.2.1. Analysis of the 1-Dimensional Situation. Recall that in the 1-dimensional case $\pi = \pi^1$ coincides with $\lambda = \lambda^1$.

To analyze the 1-dimensional situation, let $X := (x^i)_{i=1}^n$ be the given point configuration in $[0, 1)$. Without loss of generality we assume that $0 \leq x^1 < \dots < x^n < 1$. Since $\delta^*(1) = 0$ we do not need to consider the whole grid $\bar{\Gamma}(X)$ but can restrict ourselves to the set $\Gamma(X) = \{x^1, \dots, x^n\}$. For the same reason, let us set $y^+ := x^1$ if $y > x^n$ (recall that, following the description given in Section 3.1, we set $y^- := x^n$ for $y < x^1$ anyhow).

As discussed above, we take points randomly from $\Gamma(X)$, but instead of using equal probability weights on $\Gamma(X)$, we use the probability distributions $w_u = w_u^{sn}$ and $w_l = w_l^{sn}$ on $\Gamma(X)$ to maximize our objective functions δ and $\bar{\delta}$, respectively. If we put $x^0 := x^n - 1$ and $x^{n+1} := x^1 + 1$, then the corresponding probability weights for δ and $\bar{\delta}$ are given by $w_l(x^i) := x^i - x^{i-1}$ and $w_u(x^i) := x^{i+1} - x^i$, respectively.

In the next lemma we will prove the following statements rigorously: If one wants to sample a point $\tau \in \Gamma(X)$ with $\delta(\tau)$ as large as possible or if one wants to enlarge the chances to sample the point τ where δ takes its maximum, its preferable to use the weights w_l instead of the equal weights $1/n$ on $\Gamma(X)$. Similarly, it is preferable to employ the weights $w_u(x^i)$, $i = 1, \dots, n$, instead of equal weights if one wants to increase the expectation of $\bar{\delta}$ or the chances of sampling the maximum of $\bar{\delta}$.

LEMMA 5.3. *Let $d = 1$ and $\tau, \bar{\tau} \in \Gamma(X)$ with $\delta(\tau) = \sup_{z \in [0, 1]} \delta(z)$ and $\bar{\delta}(\bar{\tau}) =$*

$\sup_{z \in [0,1]} \bar{\delta}(z)$. Then we have

$$w_l(\tau) \geq 1/n \quad \text{and} \quad w_u(\bar{\tau}) \geq 1/n. \quad (5.7)$$

Furthermore, let \mathbb{E} , \mathbb{E}_l , and \mathbb{E}_u denote the expectations with respect to the uniform weights $\{1/n\}$, the weights $\{w_l(x^i)\}$, and the weights $\{w_u(x^i)\}$ on the probability space $\Gamma(X)$, respectively. Then $\mathbb{E}_l(\delta) \geq \mathbb{E}(\delta)$ and $\mathbb{E}_u(\bar{\delta}) \geq \mathbb{E}(\bar{\delta})$.

Proof. Let $\nu \in [n]$ with $\tau = x^\nu$. Assume first $w_l(x^\nu) < 1/n$, i.e., $x^\nu - x^{\nu-1} < 1/n$. If $\nu > 1$, then

$$\delta(x^{\nu-1}) = x^{\nu-1} - \frac{\nu-2}{n} > x^\nu - \frac{\nu-1}{n} = \delta(x^\nu).$$

If $\nu = 1$, then however

$$\delta(x^n) = x^n - \frac{n-1}{n} = x^0 + \frac{1}{n} > x^1 = \delta(x^\nu).$$

So both cases result in a contradiction.

We prove now $\mathbb{E}_l(\delta) \geq \mathbb{E}(\delta)$ by induction over the cardinality n of X . For $n = 1$ we trivially have $\mathbb{E}_l(\delta) = x^1 = \mathbb{E}(\delta)$.

Therefore, let the statement be true for n and consider an ordered set $\Gamma(X) := \{x^1, \dots, x^{n+1}\}$. Let δ achieve its maximum in $x^\nu \in \Gamma(X)$. We already proved that $w_l(x^\nu) = x^\nu - x^{\nu-1} \geq 1/(n+1)$ holds. With the notation

$$\tilde{x}^i := x^i \quad \text{if } 1 \leq i < \nu, \quad \tilde{x}^i := x^{i+1} - \frac{1}{n+1} \quad \text{if } i \geq \nu,$$

and

$$\hat{x}^i := \frac{n+1}{n} \tilde{x}^i, \quad i = 1, \dots, n, \quad \text{and} \quad \hat{x}^0 := \hat{x}^n - 1,$$

we get

$$\begin{aligned} \mathbb{E}_l(\delta) &= \sum_{i=1}^{n+1} w_l(x^i) \delta(x^i) = \frac{\delta(x^\nu)}{n+1} + \left(w_l(x^\nu) - \frac{1}{n+1} \right) \delta(x^\nu) + \sum_{\substack{i=1 \\ i \neq \nu}}^{n+1} w_l(x^i) \delta(x^i) \\ &\geq \frac{\delta(x^\nu)}{n+1} + \left(w_l(x^{\nu+1}) + w_l(x^\nu) - \frac{1}{n+1} \right) \delta(x^{\nu+1}) + \sum_{\substack{i=1 \\ i \notin \{\nu, \nu+1\}}}^{n+1} w_l(x^i) \delta(x^i) \\ &= \frac{\delta(x^\nu)}{n+1} + \left(\tilde{x}^1 - \tilde{x}^n + \frac{n}{n+1} \right) \tilde{x}^1 + \sum_{i=2}^n (\tilde{x}^i - \tilde{x}^{i-1}) \left(\tilde{x}^i - \frac{i-1}{n+1} \right) \\ &= \frac{\delta(x^\nu)}{n+1} + \left(\frac{n}{n+1} \right)^2 \sum_{i=1}^n (\hat{x}^i - \hat{x}^{i-1}) \left(\hat{x}^i - \frac{i-1}{n} \right). \end{aligned}$$

On the other hand we have

$$\begin{aligned} \mathbb{E}(\delta) &= \sum_{i=1}^{n+1} \frac{1}{n+1} \delta(x^i) = \frac{\delta(x^\nu)}{n+1} + \sum_{i=1}^n \frac{1}{n+1} \left(\tilde{x}^i - \frac{i-1}{n+1} \right) \\ &= \frac{\delta(x^\nu)}{n+1} + \left(\frac{n}{n+1} \right)^2 \sum_{i=1}^n \frac{1}{n} \left(\hat{x}^i - \frac{i-1}{n} \right). \end{aligned}$$

These calculations and our induction hypothesis, applied to $\{\hat{x}^1, \dots, \hat{x}^n\}$, lead to $\mathbb{E}_l(\delta) \geq \mathbb{E}(\delta)$. For $\mu \in [n]$ with $\bar{\tau} = x^\mu$ the inequalities $w_u(x^\mu) \geq 1/n$ and $\mathbb{E}_u(\bar{\delta}) \geq \mathbb{E}(\bar{\delta})$ can be proved in a similar manner. \square

to fulfill $\sigma_\nu(X) = \tau$. Thus, the probability of ι being a $\delta(X)$ -critical index is just the number of k -tuples $(\sigma_1, \dots, \sigma_k)$ of permutations fulfilling (5.8), divided by $(n!)^k$.

For given pairwise distinct values $\sigma_1(i_1), \dots, \sigma_k(i_k)$ the j -th condition of (5.8) is satisfied by $((i_j - 1) \dots (i_j - (k - 1)))(n - k)!$ permutations σ_j . Since all k conditions in (5.8) can be solved independently of each other, it is now easy to deduce the statement of the Proposition. \square

To state the corresponding proposition for $\bar{\delta}$, we have to introduce *Stirling numbers of second kind* $S(d, k)$. For $k \in \mathbb{N}$, $k \leq d$ let $S(d, k)$ denote the number of partitions of $[d]$ into k non-empty subsets. A closed formula for $S(d, k)$ is

$$S(d, k) = \sum_{j=0}^k \frac{(-1)^j (k - j)^d}{j! (k - j)!}.$$

This formula and other useful identities can, e.g., be found in [35].

PROPOSITION 5.5. *Let X be a uniformly distributed random variable in $[0, 1]^{nd}$. Let $\iota = (i_1, \dots, i_d) \in [n]^d$. Then ι is a $\bar{\delta}(X)$ -critical multi-index with probability*

$$\sum_{k=1}^d S(d, k) \left(\frac{(n - k)!}{n!} \right)^{d-1} \prod_{j=1}^d \left(\prod_{\nu=1}^{k-1} (i_j - \nu) \right).$$

Proof. We just need to consider the case where the almost-sure event $|\Gamma_j(X)| = n$ for all $j \in [d]$ holds. For $j = 1, \dots, d$ let $\sigma_j := \sigma_j(X) : [n] \rightarrow [n]$ be the permutation with

$$x_j^{\sigma_j(1)} < x_j^{\sigma_j(2)} < \dots < x_j^{\sigma_j(n)} < 1.$$

Then $\Phi(\iota)_j = \phi_j(i_j) = x_j^{\sigma_j(i_j)}$ for all $j \in [d]$. It is easy to see that the surface $\bar{S}_j(\Phi(\iota))$ is $\bar{\delta}(X)$ -critical if and only if the condition

$$\forall j \in [d] \forall l \in [d] \setminus \{j\} : x_l^{\sigma_j(i_j)} \leq \Phi(\iota)_l = x_l^{\sigma_l(i_l)}$$

is satisfied. This can be rewritten as

$$\begin{aligned} \sigma_1^{-1}(\sigma_2(i_2)), \sigma_1^{-1}(\sigma_3(i_3)), \dots, \sigma_1^{-1}(\sigma_d(i_d)) &\leq i_1, \\ \sigma_2^{-1}(\sigma_1(i_1)), \sigma_2^{-1}(\sigma_3(i_3)), \dots, \sigma_2^{-1}(\sigma_d(i_d)) &\leq i_2, \\ \vdots &\vdots \\ \sigma_d^{-1}(\sigma_1(i_1)), \sigma_d^{-1}(\sigma_2(i_2)), \dots, \sigma_d^{-1}(\sigma_{d-1}(i_{d-1})) &\leq i_d. \end{aligned} \tag{5.9}$$

If $|\{\sigma_1(i_1), \dots, \sigma_d(i_d)\}| = k$, then there exist

$$S(d, k) n! ((n - k)!)^{d-1} \prod_{j=1}^d \prod_{\nu=1}^{k-1} (i_j - \nu)$$

permutations satisfying (5.9). With this observation and the fact that all components x_j^i , $i \in [n]$, $j \in [d]$, of X are stochastically independent, it is now easy to deduce the statement of Proposition 5.5. \square

6. Experimental Results. We now present the experimental evaluations of the algorithms. We will compare our basic and improved algorithms, `TA_basic` and `TA_improved`, against the algorithm of Winker and Fang [43], and also give a brief comparison against the genetic algorithm of Shah [36] and the integer programming-based algorithm of Thiémarc [41].

6.1. Experimental Setup. We divide our experiments into a thorough comparison against the algorithm of Winker and Fang [43], given in Section 6.3, and more brief comparisons against the algorithms of Shah [36] and Thiémond [41], in Sections 6.4 and 6.5, respectively. The algorithms `TA_basic` and `TA_improved`, as well as the algorithm of Winker and Fang [43], were implemented by the authors in the C programming language, based on the code used in [44]. All implementations were done with equal care. In the case of Winker and Fang [43], while we did have access to the original Fortran source code (thanks to P. Winker), due to lack of compatible libraries we could not use it, and were forced to do a re-implementation.

For the integer programming-based algorithm of Thiémond [41], E. Thiémond has kindly provided us use of the source code. This source code was modified only as far as necessary for compatibility with newer software versions – specifically, we use version 11 of the CPLEX integer programming package, while the code of Thiémond was written for an older version. Finally, Shah has provided us with the application used in the experiments of [36], but as this application is hard-coded to use certain types of point sets only, we restrict ourselves to comparing with the experimental data published in [36]. Random numbers were generated using the Gnu C library pseudo-random number generator.

The instances used in the experiments are described in Section 6.2. For some instances, we are able to compute the exact discrepancy values either using an implementation of the algorithm of Dobkin et al. [8], available from the third author’s homepage³, or via the integer programming-based algorithm of Thiémond [41]. These algorithms both have far better time dependency than that of Bundschuh and Zhu [3], allowing us to report exact data for larger instances than previously done. For those instances where this is too costly, we report instead the largest discrepancy value found by any algorithm in any trial; these imprecise values are marked by a star. Note that this includes some trials with other (more time-consuming) parameter settings than those of our published experiments; thus sometimes, none of the reported algorithms are able to match the approximate max value.

As parameter settings for the neighbourhood for `TA_basic`, we use $\ell = \lfloor \frac{n}{8} \rfloor$ if $n \geq 100$, and $\ell = \lfloor \frac{n}{4} \rfloor$ otherwise, and $mc = 2$ throughout. These settings showed reasonable performance in our experiments and in [44]. For `TA_improved`, these parameters are handled by the scaling described in Section 4.2.

Throughout, for our algorithms and for the Winker and Fang algorithm, we estimate the expected outcome of running 10 independent trials of 100,000 iterations each and returning the largest discrepancy value found, and call this the *best-of-10 value*. The estimation is computed from a basis of 100 independent trials, as suggested by Johnson [27], which strongly decreases irregularities due to randomness compared to the method of taking 10 independent best-of-10 values and averaging these. The comparisons are based on a fix number of iterations, rather than equal running times, as the point of this paper is to compare the strengths of the involved concepts and ideas, rather than implementation tweaks. For this purpose, using a re-implementation rather than the original algorithm of Winker and Fang [43] has the advantage that all algorithms compared use the same code base, compiler, and libraries, including the choice of pseudo-random number generator. This further removes differences that are not interesting to us.

³<http://www.mpi-inf.mpg.de/~wahl/>.

| Algorithm | Smaller instance $d = 10, n = 100$ | Larger instance $d = 20, n = 1000$ |
|----------------------------------|---------------------------------------|---------------------------------------|
| TA_basic | 0.78s | 9.34s |
| TA_improved, δ only | 1.22s | 10.94s |
| TA_improved, $\bar{\delta}$ only | 0.85s | 9.11s |
| TA_improved, mixed form | 1.87s | 20.37s |
| Winker & Fang | 0.61s | 7.2s |

TABLE 6.1

Running times for the considered algorithms. All algorithms executed one trial of 100,000 iterations. The inputs are two randomly generated point sets.

6.2. Instances. Our point sets are of four types: Halton sequences [22], Faure sequences [15], Sobol’ point sets [39], and so-called Good Lattice Points (GLP) [43]. The Halton sequences and GLPs were generated by programs written by the authors, the Faure sequences by a program of John Burkardt [4], and the Sobol’ sequences using the data and code of Stephen Joe and Frances Kuo [26, 29].

6.3. Comparisons against the Algorithm of Winker and Fang. To begin the comparisons, an indication of the running times of the algorithms is given in Table 6.1. As can be seen from the table, TA_basic takes slightly more time than our implementation of Winker and Fang, and TA_improved takes between two and three times as long, mainly due to the snapping procedures. For TA_improved, we report the separate times for δ and $\bar{\delta}$ optimization, as well as the time required for a mixed optimization of both (as is done in TA_basic). As can be seen, the overhead due to splitting is negligible to non-existent.

The parameter settings for our implementation of the algorithm of Winker and Fang are as follows. Since our experiments did not reveal a strong influence of the choice of α on the quality of the algorithm, we fix $\alpha := 0.995$ for our experiments. Winker and Fang do not explicitly give a rule how one should choose k and mc . For the small-dimensional data (dimensions 4 to 11), we use the settings of [43]. For the other tests, we use $mc = 3$ if $d \leq 12$ and $mc = 4$ otherwise, and $k = 41$ if $n \leq 500$ and $k = 301$ otherwise. This seems to be in line with the choices of Winker and Fang for the sizes used.

For all GLP sets tested in [43] all algorithms behave reasonably, with both of our algorithms generally outperforming our implementation of Winker and Fang, and with TA_improved showing much higher precision than TA_basic. Here we omit a full presentation of the results, which, nevertheless, can be found in the extended preprint version [21]. We only note that our re-implementation of the Winker and Fang algorithm gives notably worse results than what was reported in [43] for the same instances, in a way that cannot be fully explained by parameter settings. After significant experimentation, the best hypothesis we can provide is that there might be a difference in the behavior of the pseudo-random number generators used (in particular, as [43] uses a random number library we do not have access to). Still, even compared to the results reported in [43], our algorithms, and TA_improved in particular, still fare well.

Table 6.2 shows the new data, for larger-scale instances. A few trends are noticeable, in particular for the higher-dimensional data. Here, the algorithm of Winker and Fang seems to deteriorate, and there is also a larger difference emerging between TA_basic and TA_improved, in particular for the Sobol’ sets. However, as can be seen for the 2048-point, 20-dimensional Sobol’ set, it does happen that the lower bound is quite imprecise. (The value of 0.0724 for this point set was discovered only a handful

| Name | d | n | $d_{\infty}^*(\cdot)$ found | TA.basic | | TA.improved | | Winker & Fang | |
|--------|-----|------|--------------------------------|----------|------------|-------------|------------|---------------|------------|
| | | | | Hits | Best-of-10 | Hits | Best-of-10 | Hits | Best-of-10 |
| Sobol' | 7 | 256 | 0.0883 | 1 | 0.0804 | 78 | 0.0883 | 0 | 0.0819 |
| Sobol' | 7 | 512 | 0.0452 | 1 | 0.0440 | 17 | 0.0451 | 0 | 0.0395 |
| Sobol' | 8 | 128 | 0.1202 | 0 | 0.1198 | 98 | 0.1202 | 0 | 0.1102 |
| Sobol' | 9 | 128 | 0.1372 | 8 | 0.1367 | 100 | 0.1372 | 0 | 0.1254 |
| Sobol' | 10 | 128 | 0.1787 | 36 | 0.1787 | 100 | 0.1787 | 0 | 0.1606 |
| Sobol' | 11 | 128 | 0.1811 | 14 | 0.1811 | 97 | 0.1811 | 0 | 0.1563 |
| Sobol' | 12 | 128 | 0.1885 | 1 | 0.1873 | 82 | 0.1885 | 0 | 0.1689 |
| Sobol' | 12 | 256 | 0.1110* | 2 | 0.1108 | 41 | 0.1110 | 0 | 0.0908 |
| Faure | 7 | 343 | 0.1298 | 21 | 0.1297 | 100 | 0.1298 | 0 | 0.1143 |
| Faure | 8 | 121 | 0.1702 | 99 | 0.1702 | 100 | 0.1702 | 0 | 0.1573 |
| Faure | 9 | 121 | 0.2121 | 98 | 0.2121 | 100 | 0.2121 | 0 | 0.1959 |
| Faure | 10 | 121 | 0.2574 | 95 | 0.2574 | 100 | 0.2574 | 0 | 0.2356 |
| Faure | 11 | 121 | 0.3010 | 100 | 0.3010 | 100 | 0.3010 | 0 | 0.2632 |
| Faure | 12 | 169 | 0.2718 | 73 | 0.2718 | 100 | 0.2718 | 0 | 0.1708 |
| GLP | 6 | 343 | 0.0870 | 1 | 0.0869 | 36 | 0.0870 | 0 | 0.0778 |
| GLP | 7 | 343 | 0.0888 | 3 | 0.0883 | 28 | 0.0888 | 0 | 0.0791 |
| GLP | 8 | 113 | 0.1422 | 6 | 0.1399 | 95 | 0.1422 | 0 | 0.1303 |
| GLP | 9 | 113 | 0.1641 | 98 | 0.1641 | 100 | 0.1641 | 0 | 0.1490 |
| GLP | 10 | 113 | 0.1871 | 1 | 0.1862 | 94 | 0.1871 | 0 | 0.1744 |
| Sobol' | 20 | 128 | 0.2616* | 0 | 0.2576 | 51 | 0.2616 | 0 | 0.0497 |
| Sobol' | 20 | 256 | 0.1856* | 13 | 0.1854 | 49 | 0.1856 | 0 | 0.0980 |
| Sobol' | 20 | 512 | 0.1336* | 0 | 0.1080 | 86 | 0.1336 | 0 | 0.0635 |
| Sobol' | 20 | 1024 | 0.1349* | 0 | 0.0951 | 0 | 0.1330 | 0 | 0.0560 |
| Sobol' | 20 | 2048 | 0.0724* | 0 | 0.0465 | 0 | 0.0505 | 0 | 0.0370 |
| Faure | 20 | 529 | 0.2615* | 0 | 0.2587 | 98 | 0.2615 | 0 | 0.0275 |
| Faure | 20 | 1500 | 0.0740* | 0 | 0.0733 | 14 | 0.0740 | 0 | 0.0347 |
| GLP | 20 | 149 | 0.2581* | 1 | 0.2548 | 65 | 0.2581 | 0 | 0.0837 |
| GLP | 20 | 227 | 0.1902* | 0 | 0.1897 | 1 | 0.1899 | 0 | 0.0601 |
| GLP | 20 | 457 | 0.1298* | 0 | 0.1220 | 3 | 0.1272 | 0 | 0.0519 |
| GLP | 20 | 911 | 0.1013* | 0 | 0.0975 | 8 | 0.1013 | 0 | 0.0315 |
| GLP | 20 | 1619 | 0.0844* | 0 | 0.0809 | 2 | 0.0844 | 0 | 0.0299 |
| Sobol' | 50 | 2000 | 0.1030* | 0 | 0.0952 | 0 | 0.1024 | 0 | 0.0005 |
| Sobol' | 50 | 4000 | 0.0677* | 0 | 0.0597 | 0 | 0.0665 | 0 | 0.00025 |
| Faure | 50 | 2000 | 0.3112* | 0 | 0.2868 | 100 | 0.3112 | 0 | 0.0123 |
| Faure | 50 | 4000 | 0.1979* | 0 | 0.1912 | 0 | 0.1978 | 0 | 0.0059 |
| GLP | 50 | 2000 | 0.1465* | 0 | 0.1317 | 0 | 0.1450 | 0 | 0.0005 |
| GLP | 50 | 4000 | 0.1205* | 0 | 0.1053 | 0 | 0.1201 | 0 | 0.0003 |

TABLE 6.2

New instance comparisons. Discrepancy values marked with a star are lower bounds only (i.e., largest discrepancy found over all executions of algorithm variants). All data is computed using 100 trials of 100,000 iterations; reported is the average value of best-of-10 calls, and number of times (out of 100) that the optimum (or a value matching the largest known value) was found.

of times over nearly 5000 trials of algorithm variants and settings.)

The highest-dimensional sets ($d = 50$) illustrate the deterioration of Winker and Fang with increasing dimension; for many of the settings, the largest error this algorithm finds is exactly $1/n$ (due to the zero-volume box containing the origin with one point).

6.4. Comparisons with the Algorithm by Shah. Table 6.3 lists the point sets used by Shah [36]. The Faure sets here are somewhat nonstandard in that they exclude the origin, i.e., they consist of points 2 through $n + 1$ of the Faure sequence, where the order of the points is as produced by the program of Burkhardt [4].

Some very small point sets were omitted, as every reported algorithm would find the optimum every time. For all but one of the point sets, the exact discrepancy could be computed; the remaining instance is the first 500 points of the 10-dimensional Faure

| Class | d | n | $d_{\infty}^*(\cdot)$ | TA_basic | | TA_improved | | Shah | |
|--------|-----|------|-----------------------|----------|------------|-------------|------------|------------------|-----------------------|
| | | | | Hits | Best-of-10 | Hits | Best-of-10 | Hits | Best Found |
| Halton | 5 | 50 | 0.1886 | 100 | 0.1886 | 100 | 0.1886 | 81 | 0.1886 |
| Halton | 7 | 50 | 0.2678 | 100 | 0.2678 | 100 | 0.2678 | 22 | 0.2678 |
| Halton | 7 | 100 | 0.1714 | 9 | 0.1710 | 100 | 0.1714 | 13 | 0.1714 |
| Halton | 7 | 1000 | 0.0430 | 0 | 0.0424 | 81 | 0.0430 | 8 ⁽¹⁾ | 0.0430 ⁽¹⁾ |
| Faure | 10 | 50 | 0.4680 | 100 | 0.4680 | 100 | 0.4680 | 97 | 0.4680 |
| Faure | 10 | 100 | 0.2483 | 52 | 0.2483 | 100 | 0.2483 | 28 | 0.2483 |
| Faure | 10 | 500 | 0.0717* | 2 | 0.0701 | 100 | 0.0717 | 0 ⁽¹⁾ | 0.0689 ⁽¹⁾ |

TABLE 6.3

Comparison against point sets used by Shah. Reporting average value of best-of-10 calls, and number of times (out of 100) that the optimum was found; for Shah, reporting highest value found, and number of times (out of 100) this value was produced. The discrepancy value marked with a star is lower bound only (i.e., largest value found by any algorithm). Values marked (1) are recomputed using the same settings as in [36].

| Instance | TA_improved | | Thiémard: Initial | | Same time, result | Same result, time |
|----------------|-------------|--------|-------------------|--------|-------------------|-------------------|
| | Time | Result | Time | Result | | |
| Faure-12-169 | 25s | 0.2718 | 1s | 0.2718 | 0.2718 | 1s |
| Sobol'-12-128 | 20s | 0.1885 | 1s | 0.1463 | 0.1463 | 453s (7.6m) |
| Sobol'-12-256 | 35s | 0.1110 | 3s | 0.0872 | 0.0872 | 1.6 days |
| Faure-20-1500 | 280s (4.7m) | 0.0740 | 422s (7m) | 0.0732 | None | > 4 days |
| GLP-20-1619 | 310s (5.2m) | 0.0844 | 564s (9.4m) | 0.0572 | None | > 5 days |
| Sobol'-50-4000 | 2600s (42m) | 0.0665 | 32751s (9h) | 0.0743 | None | 32751s (9h) |
| GLP-50-4000 | 2500s (43m) | 0.1201 | 31046s (8.6h) | 0.0301 | None | > 5 days |

TABLE 6.4

Comparison against the integer programming-based algorithm of Thiémard [41]. The values for TA_improved represent the time and average result of a best-of-10 computation with 100,000 iterations per trial. The middle pair of columns give the time required for [41] to return a first output, and the value of this output; the last two columns report the lower bound reached by [41] if allocated the same time that TA_improved needs for completion, and the time required by [41] to match the result of TA_improved.

sequence.

Most of the sets seem too easy to really test the algorithms, i.e., all variants frequently find essentially optimal points. The one exception is the last item, which shows a clear advantage for our algorithms. We also find (again) that TA_improved has a better precision than the other algorithms.

6.5. Comparisons with the Algorithms by Thiémard. Finally, we give a quick comparison against the integer programming-based algorithm of Thiémard [41]. Since [41] has the feature that running it for a longer time produces gradually stronger bounds, we report three different checkpoint values; see Table 6.4 for details. The results are somewhat irregular; however, [41] may require a lot of time to report a first value, and frequently will not improve significantly on this initial lower bound except after very large amounts of computation time (for example, for the 12-dimensional, 256-point Sobol' set, the value 0.0872 is discovered in seconds, while the first real improvement takes over an hour to produce).

Thiémard also constructed a second algorithm for discrepancy estimation, based on delta-covers [40]; this is freely downloadable from Thiémard's homepage. Its prime feature is that it provides upper bounds with a non-trivial running time guarantee. The lower bounds that it produces are not as helpful as the upper bounds, e.g., it was reported in [11] and [36] that the lower bounds from the preliminary version of

TA_basic [44] and the genetic algorithm of Shah [36] are better. Thus we omit this kind of comparison here.

7. Conclusion. Our numerical experiments clearly indicate that the improvements made from the algorithm of Winker and Fang in TA_basic and TA_improved greatly increases the quality of the lower bounds, in particular for the difficult higher-dimensional problem instances. Also the comparison with the algorithms by Thiéard and Shah, respectively, is favorable for our algorithms. Thus we conclude that the algorithms TA_basic and TA_improved presented in the current work represent significant improvements over previous lower-bound heuristics for computing the star discrepancy, and to the best of our knowledge, make up the best performing star discrepancy estimation algorithms in higher dimension.

Acknowledgments. We gratefully acknowledge Manan Shah, Eric Thiéard, and Peter Winker for providing us with source code and implementations of the applications used in their experiments (in [36], [41], and [43], respectively), and in general for helpful comments.

Michael Gnewuch was supported by the German Research Foundation (DFG) under grants GN 91/3-1 and GN 91/4-1. Part of his work was done while he was at Columbia University in the City of New York.

Magnus Wahlström is supported by the DFG via its priority program "SPP 1307: Algorithm Engineering" under grant DO 749/4-1.

Carola Winzen is a recipient of the Google Europe Fellowship in Randomized Algorithms, and this research is supported in part by this Google Fellowship.

REFERENCES

- [1] I. ALTHÖFER AND K.-U. KOSCHNICK, *On the Convergence of "Threshold Accepting"*, Applied Mathematics and Optimization, 24 (1991), pp. 183–195.
- [2] J. BECK AND W. W. L. CHEN, *Irregularities of Distribution*, Cambridge University Press, Cambridge, 1987.
- [3] 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), pp. 115–133.
- [4] J. BURKARDT, *FAURE – the Faure quasirandom sequence*. http://people.sc.fsu.edu/~jburkardt/m_src/faure/faure.html.
- [5] B. CHAZELLE, *The Discrepancy Method*, Cambridge University Press, Cambridge, 2000.
- [6] J. DICK, G. LEOBACHER, AND F. PILLICHSHAMMER, *Construction algorithms for digital nets with low weighted star discrepancy*, SIAM J. Numer. Anal., 43 (2005), pp. 76–95.
- [7] J. DICK AND F. PILLICHSHAMMER, *Digital Nets and Sequences*, Cambridge University Press, Cambridge, 2010.
- [8] D. P. DOBKIN, D. EPPSTEIN, AND D. P. MITCHELL, *Computing the discrepancy with applications to supersampling patterns*, ACM Trans. Graph., 15 (1996), pp. 354–376.
- [9] B. DOERR, M. GNEWUCH, P. KRITZER, AND F. PILLICHSHAMMER, *Component-by-component construction of low-discrepancy point sets of small size*, Monte Carlo Methods Appl., 14 (2008), pp. 129–149.
- [10] B. DOERR, M. GNEWUCH, AND M. WAHLSTRÖM, *Implementation of a component-by-component algorithm to generate low-discrepancy samples*, in Monte Carlo and Quasi-Monte Carlo Methods 2008, P. L’Ecuyer and A. B. Owen, eds., Berlin Heidelberg, 2009, Springer.
- [11] ———, *Algorithmic construction of low-discrepancy point sets via dependent randomized rounding*, J. Complexity, 26 (2010), pp. 490–507.
- [12] M. DRMOTA AND R. F. TICHY, *Sequences, Discrepancies and Applications*, vol. 1651 of Lecture Notes in Mathematics, Springer, Berlin and Heidelberg, 1997.
- [13] G. DUECK AND T. SCHEUER, *Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing*, J. Comput. Phys., 90 (1990), pp. 161–175.
- [14] K. T. FANG AND Y. WANG, *Applications of Number Theoretic Methods in Statistics*, Chapman and Hall, London, 1994.

- [15] H. FAURE, *Discrepancy of sequences associated with a number system (in dimension s)*, Acta. Arith, 41 (1982), pp. 337–351. In French.
- [16] K. FRANK AND S. HEINRICH, *Computing discrepancies of Smolyak quadrature rules*, J. Complexity, 12 (1996), pp. 287–314.
- [17] P. GIANNOPOULUS, C. KNAUER, M. WAHLSTRÖM, AND D. WERNER, *Hardness of discrepancy computation and epsilon-net verification in high dimension*. Preprint, 2011. [arXiv:1103.4503 \[cs.CG\]](https://arxiv.org/abs/1103.4503).
- [18] M. GNEWUCH, *Bracketing numbers for axis-parallel boxes and applications to geometric discrepancy*, J. Complexity, 24 (2008), pp. 154–172.
- [19] ———, *Weighted geometric discrepancies and numerical integration on reproducing kernel Hilbert spaces*. J. Complexity, 2011. (doi:10.1016/j.jco.2011.02.003).
- [20] 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 (2009), pp. 115–127.
- [21] M. GNEWUCH, M. WAHLSTRÖM, AND C. WINZEN, *A new randomized algorithm to approximate the star discrepancy based on threshold accepting*. Enhanced preprint version, 2011. [arXiv:1103.2102v2 \[cs.DS\]](https://arxiv.org/abs/1103.2102v2).
- [22] J. H. HALTON, *On the efficiency of certain quasi-random sequences of points in evaluating multidimensional integrals*, Numer. Math., 2 (1960), pp. 84–90.
- [23] S. HEINRICH, *Efficient algorithms for computing the L_2 discrepancy*, Math. Comp., 65 (1996), pp. 1621–1633.
- [24] A. HINRICHS, F. PILLICHSHAMMER, AND W. CH. SCHMID, *Tractability properties of the weighted star discrepancy*, J. Complexity, 24 (2008), pp. 134–143.
- [25] S. JOE, *Construction of good rank-1 lattice rules based on the weighted star discrepancy*, in Monte Carlo and Quasi-Monte Carlo Methods 2004, H. Niederreiter and D. Talay, eds., Berlin, 2006, Springer, pp. 181–196.
- [26] S. JOE AND F. Y. KUO, *Constructing Sobol’ sequences with better two-dimensional projections*, SIAM J. Sci. Comput., 30 (2008), pp. 2635–2654.
- [27] D. S. JOHNSON, *A theoretician’s guide to the experimental analysis of algorithms*, in Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges, M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, eds., American Mathematical Society, Providence, 2002, pp. 215–250.
- [28] S. KIRCKPATRICK, C. GELATT, AND M. VECCHI, *Optimization by simulated annealing*, Science, 20 (1983), pp. 671–680.
- [29] F. Y. KUO, *Sobol sequence generator*, 2010. <http://web.maths.unsw.edu.au/~fkuo/sobol/index.html>.
- [30] C. LEMIEUX, *Monte Carlo and Quasi-Monte Carlo Sampling*, Springer, New York, 2009.
- [31] J. MATOUŠEK, *Geometric Discrepancy*, Springer, Berlin, 2nd ed., 2009.
- [32] H. NIEDERREITER, *Discrepancy and convex programming*, Ann. Mat. Pura Appl., 93 (1972), pp. 89–97.
- [33] ———, *Random Number Generation and Quasi-Monte Carlo Methods*, vol. 63 of SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, 1992.
- [34] E. NOVAK AND H. WOŹNIAKOWSKI, *Tractability of Multivariate Problems. Vol. 2: Standard Information for Functionals*, EMS Tracts in Mathematics, European Mathematical Society (EMS), Zürich, 2010.
- [35] J. RIORDAN, *An Introduction to Combinatorial Analysis*, Wiley, New York, 1958.
- [36] M. SHAH, *A genetic algorithm approach to estimate lower bounds of the star discrepancy*, Monte Carlo Methods Appl., 16 (2010), pp. 379–398.
- [37] V. SINESCU AND S. JOE, *Good lattice rules based on the general weighted star discrepancy*, Math. Comp., 76 (2007), pp. 989–1004.
- [38] I. H. SLOAN, *On the unreasonable effectiveness of QMC*. Slides of a plenary talk at the 9th International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, 2010. <http://mcqmc.mimuw.edu.pl/?page=presentations>.
- [39] I.M. SOBOLOV, *The distribution of points in a cube and the approximate evaluation of integrals*, Zh. Vychisl. Mat. i Mat. Fiz., 7 (1967), pp. 784–802. In Russian.
- [40] E. THIÉMARD, *An algorithm to compute bounds for the star discrepancy*, J. Complexity, 17 (2001), pp. 850–880.
- [41] ———, *Optimal volume subintervals with k points and star discrepancy via integer programming*, Math. Meth. Oper. Res., 54 (2001), pp. 21–45.
- [42] T. T. WARNOCK, *Computational investigations of low-discrepancy point sets*, in Applications of number theory to numerical analysis, S. K. Zaremba, ed., New York, 1972, Academic Press, pp. 319–343.

- [43] P. WINKER AND K. T. FANG, *Applications of threshold-accepting to the evaluation of the discrepancy of a set of points*, SIAM J. Numer. Anal., 34 (1997), pp. 2028–2042.
- [44] C. WINZEN, *Approximative Berechnung der Sterndiskrepanz*, Diplomarbeit, Mathematisches Seminar, Christian-Albrechts-Universität zu Kiel, Kiel, 2007. Also appeared in: VDM Verlag Dr. Müller. ISBN: 9783639275254.

Appendix A. Calculation of $\lambda^d(A_\varepsilon(z))$ and $\pi^d(A_\varepsilon(z))$.

LEMMA A.1. *Let $\varepsilon \in (0, 1]$, and let $z \in [0, 1]^d$ with $V_z \geq \varepsilon$. Then*

$$\lambda^d(A_\varepsilon(z)) = V_z - (V_z - \varepsilon) \sum_{k=0}^{d-1} \frac{(-\ln(1 - \varepsilon/V_z))^k}{k!}. \quad (\text{A.1})$$

Proof. Let $V_z \geq \varepsilon$. Then we have

$$\lambda^d(A_\varepsilon(z)) = \int_{\alpha_1}^{z_1} \dots \int_{\alpha_d}^{z_d} d\zeta_d \dots d\zeta_1,$$

where

$$\alpha_1 = \frac{V_z - \varepsilon}{z_2 z_3 \dots z_d}, \quad \alpha_2 = \frac{V_z - \varepsilon}{\zeta_1 z_3 \dots z_d}, \dots, \quad \alpha_d = \frac{V_z - \varepsilon}{\zeta_1 \zeta_2 \dots \zeta_{d-1}}.$$

We prove formula (A.1) by induction over the dimension d . If $d = 1$, then clearly $\lambda(A_\varepsilon(z)) = \varepsilon$. Let now $d \geq 2$. We denote by \tilde{z} the $(d-1)$ -dimensional vector (z_2, \dots, z_d) and by $\tilde{\varepsilon}$ the term $(\varepsilon + (\zeta_1 - z_1)V_{\tilde{z}})/\zeta_1$. Furthermore we define for $i \in [d-1]$ the lower integration limit $\tilde{\alpha}_i = (V_{\tilde{z}} - \tilde{\varepsilon})/(\zeta_2 \dots \zeta_i \tilde{z}_{i+1} \dots \tilde{z}_{d-1})$. Note that $\tilde{\alpha}_i = \alpha_{i+1}$. Then, by our induction hypothesis,

$$\begin{aligned} \lambda^d(A_\varepsilon(z)) &= \int_{\alpha_1}^{z_1} \int_{\tilde{\alpha}_1}^{\tilde{z}_1} \dots \int_{\tilde{\alpha}_{d-1}}^{\tilde{z}_{d-1}} d\zeta_d \dots d\zeta_2 d\zeta_1 \\ &= \int_{\alpha_1}^{z_1} \left(V_{\tilde{z}} - (V_{\tilde{z}} - \tilde{\varepsilon}) \sum_{k=0}^{d-2} \frac{(-\ln(1 - \tilde{\varepsilon}/V_{\tilde{z}}))^k}{k!} \right) d\zeta_1 \\ &= V_z - (V_z - \varepsilon) - (V_z - \varepsilon) \left[\sum_{k=1}^{d-1} \frac{1}{k!} \ln \left(\frac{V_{\tilde{z}}}{V_z - \varepsilon} \zeta_1 \right)^k \right]_{\zeta_1 = \alpha_1}^{z_1} \\ &= V_z - (V_z - \varepsilon) \sum_{k=0}^{d-1} \frac{(-\ln(1 - \varepsilon/V_z))^k}{k!}. \end{aligned}$$

□

PROPOSITION A.2. *Let $d \geq 2$. For $z \in [0, 1]^d$ with $V_z > \varepsilon$, we obtain*

$$\lambda^d(A_\varepsilon(z)) = \frac{1}{d!} \frac{\varepsilon^d}{V_z^{d-1}} \sum_{k=0}^{\infty} b_k(d) \left(\frac{\varepsilon}{V_z} \right)^k$$

with positive coefficients

$$b_k(2) = \frac{2}{(k+1)(k+2)}, \quad b_k(3) = \frac{6}{(k+2)(k+3)} \sum_{\nu=0}^k \frac{1}{\nu+1}$$

and

$$b_k(d) = \frac{d!}{(k+d-1)(k+d)} \sum_{k_1=0}^k \cdots \sum_{k_{d-2}=0}^{k_{d-3}} \prod_{j=1}^{d-2} \frac{1}{k_j + d - j - 1} \quad \text{for } d \geq 4.$$

The power series converges for each $\epsilon > 0$ uniformly and absolutely for all $V_z \in [\epsilon + \epsilon, 1]$. Furthermore, we have $b_0(d) = 1$ and $b_1(d) = d(d-1)/2(d+1)$.

Proof. To prove the power series expansion, we consider the function

$$R(x, d) = 1 - (1-x) \sum_{k=0}^{d-1} \frac{(-\ln(1-x))^k}{k!} \quad \text{for } x \in [0, 1].$$

Due to Lemma A.1 we have $\lambda^d(A_\epsilon(z)) = V_z R(\epsilon/V_z, d)$. Since $\partial_x R(x, d) = (-\ln(1-x))^{d-1}/(d-1)!$, it suffices to prove the following statement by induction over d :

$$\frac{(-\ln(1-x))^{d-1}}{(d-1)!} = \frac{1}{d!} \sum_{k=0}^{\infty} (k+d)b_k(d)x^{k+d-1}, \quad (\text{A.2})$$

where the power series converges for each $\epsilon > 0$ uniformly and absolutely on $[0, 1 - \epsilon]$. Let first $d = 2$. Then

$$-\ln(1-x) = \sum_{k=1}^{\infty} \frac{x^k}{k} = \frac{1}{2!} \sum_{k=0}^{\infty} (k+2)b_k(2)x^{k+1},$$

and the required convergence of the power series is obviously given. Now let $d \geq 3$. Our induction hypothesis yields

$$\begin{aligned} \partial_x \frac{(-\ln(1-x))^{d-1}}{(d-1)!} &= \frac{1}{1-x} \frac{(-\ln(1-x))^{d-2}}{(d-2)!} \\ &= \left(\sum_{\nu=0}^{\infty} x^\nu \right) \left(\frac{1}{(d-1)!} \sum_{\mu=0}^{\infty} (\mu+d-1)b_\mu(d-1)x^{\mu+d-2} \right) \\ &= \frac{1}{d!} \sum_{k=0}^{\infty} \left(d \sum_{\mu=0}^k (\mu+d-1)b_\mu(d-1) \right) x^{k+d-2}, \end{aligned}$$

where the last power series converges as claimed above. Now

$$\begin{aligned} d \sum_{\mu=0}^k (\mu+d-1)b_\mu(d-1) &= \sum_{\mu=0}^k \frac{d!}{(\mu+d-2)} \sum_{\mu_1=0}^{\mu} \cdots \sum_{\mu_{d-3}=0}^{\mu_{d-4}} \prod_{j=2}^{d-3} \frac{1}{\mu_j + d - 2 - j} \\ &= d! \sum_{\nu_1=0}^k \sum_{\nu_2=0}^{\nu_1} \cdots \sum_{\nu_{d-2}=0}^{\nu_{d-3}} \prod_{j=1}^{d-2} \frac{1}{\nu_j + d - j - 1} \\ &= (k+d)(k+d-1)b_k(d). \end{aligned}$$

After integration we get (A.2).

Furthermore, it is easily seen that $b_0(d) = 1$ and $b_1(d) = d(d-1)/2(d+1)$. \square

We now consider the polynomial product measure π^d .

LEMMA A.3. *Let $\epsilon \in (0, 1]$, and let $z \in [0, 1]^d$ with $V_z \geq \epsilon$. Then*

$$\pi^d(A_\epsilon(z)) = V_z^d - (V_z - \epsilon)^d \sum_{k=0}^{d-1} \frac{d^k}{k!} (-\ln(1 - \epsilon/V_z))^k, \quad (\text{A.3})$$

and, as a function of V_z , $\pi^d(A_\varepsilon(z))$ is strictly increasing.

Proof. Let $V_z \geq \varepsilon$. We have

$$\pi^d(A_\varepsilon(z)) = \int_{A_\varepsilon(z)} d^d V_x^{d-1} \lambda^d(dx) = \int_0^\varepsilon G(V_z, r) dr, \quad (\text{A.4})$$

where

$$G(V_z, r) := d^d(V_z - r)^{d-1} \partial_r \lambda^d(A_r(z)).$$

From (A.1) we get for all $0 \leq r \leq \varepsilon$

$$\partial_r \lambda^d(A_r(z)) = \frac{(-\ln(1 - r/V_z))^{d-1}}{(d-1)!}.$$

If we define

$$F(r) := -(V_z - r)^d \sum_{k=0}^{d-1} \frac{d^k}{k!} (-\ln(1 - r/V_z))^k,$$

then we observe that $F'(r) = G(V_z, r)$ holds. Thus we have $\pi^d(A_\varepsilon(z)) = F(\varepsilon) - F(0)$, which proves (A.3). Furthermore, according to (A.4), we get

$$\partial_{V_z} \pi^d(A_\varepsilon(z)) = \int_0^\varepsilon \partial_{V_z} G(V_z, r) dr.$$

The integrand of the integral is positive, as the next calculation reveals:

$$\begin{aligned} \partial_{V_z} G(V_z, r) &= d^d(V_z - r)^{d-2} \frac{(-\ln(1 - r/V_z))^{d-2}}{(d-2)!} (-\ln(1 - r/V_z) - r/V_z) \\ &= d^d(V_z - r)^{d-2} \frac{(-\ln(1 - r/V_z))^{d-2}}{(d-2)!} \sum_{k=2}^{\infty} \frac{1}{k} \left(\frac{r}{V_z}\right)^k > 0 \end{aligned}$$

for all $0 < r < V_z$. Thus $\partial_{V_z} \pi^d(A_\varepsilon(z)) > 0$, and, considered as a function of V_z , $\pi^d(A_\varepsilon(z))$ is strictly increasing. \square

PROPOSITION A.4. *Let $\varepsilon \in (0, 1]$, and let $z \in [0, 1]^d$ with $V_z \geq \varepsilon$. Then we have the lower bound $\pi^d(A_\varepsilon(z)) \geq \varepsilon^d$.*

Proof. Let $V_z = \varepsilon$. Then

$$\pi^d(A_\varepsilon(z)) = \int_{[0, z]} d^d V_x^{d-1} \lambda^d(dx) = \prod_{i=1}^d z_i^d = \varepsilon^d.$$

Since $\pi^d(A_\varepsilon(z))$ is an increasing function of V_z , the lower bound holds. \square