SIAM J. NUMER. ANAL.
Vol. 50, No. 2, pp. 781–807

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), pp. 2028–2042] it is based on the optimization algorithm threshold accepting. Our improvements include, amongst others, a nonuniform 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 dimensions 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

DOI. 10.1137/110833865

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 such as optimization, combinatorics, pseudorandom number generation, option pricing, computer graphics, and other areas; see, e.g., the monographs [2, 5, 7, 12, 14, 35, 37, 39, 40].

In particular, for the important task of multivariate or infinite dimensional numerical integration, which arises frequently in fields such as finance, statistics, physics, and quantum chemistry, quasi-Monte Carlo algorithms relying on low-discrepancy samples have been studied extensively in recent 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, 26, 27, 40] 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 historically most important, and still widely used, discrepancy notion is the *star discrepancy*, which is defined as follows.

*Received by the editors May 12, 2011; accepted for publication (in revised form) January 31, 2012; published electronically April 12, 2012.

http://www.siam.org/journals/sinum/50-2/83386.html

†Institut für Informatik, Christian-Albrechts-Universität Kiel, Christian-Albrechts-Platz 4, 24098 Kiel, Germany (mig@informatik.uni-kiel.de). This author's work 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.

‡Max-Planck-Institut für Informatik, Campus E1 4, 66123 Saarbrücken, Germany (wahl@mpi-inf.mpg.de, winzen@mpi-inf.mpg.de). The second author's work was supported by the DFG via its priority program "SPP 1307: Algorithm Engineering" under grant DO 749/4-1. The third author was a recipient of the Google Europe Fellowship in Randomized Algorithms, and this research was supported in part by this Google Fellowship.

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 $[0, y) := [0, y_1) \times \cdots \times [0, y_d)$, and let V_y be the d-dimensional (Lebesgue) volume of $[0, y)$. We call

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

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

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

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

A generalization of the star discrepancy which has attracted increasing attention over the last few years is the so-called *weighted star discrepancy*; see, e.g., [6, 29, 30, 43]. In particular, it is very promising for finance applications; see [44]. 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 pseudorandom numbers are statistically independent [39], 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 one to construct low-discrepancy samples of moderate size (meaning at most polynomial in the dimension d) efficiently by generating random samples and testing whether their discrepancy is sufficiently small. Actually, there are deterministic algorithms known to construct such samples [9, 10, 11], but these exhibit high running times. The efficient calculation or approximation of the discrepancy of given point sets would lead to a practicable alternative algorithm. Efficient approximation algorithms can also be used in combination with optimization approaches to generate point sets with a low star discrepancy. A possible optimization approach described, e.g., in [8], is to apply a multidimensional optimization algorithm repeatedly to randomly jiggled versions of low discrepancy point sets to search for smaller local minima. Following the simulated annealing heuristic [33], the jiggling is gradually reduced as the algorithm proceeds.

Apart from being helpful in several applications, efficient approximation algorithms for the star discrepancy can help to shed some light on intricate theoretical problems. From the theoretical point of view, the star discrepancy is clearly the most extensively studied discrepancy measure. But despite this effort, its asymptotic behavior is unfortunately not completely understood. This open question was deemed "the great open problem of discrepancy theory" by Beck and Chen [2], and it is closely related to important problems in probability and approximation theory, namely, the improvement of the small ball inequality for the Brownian sheet and the asymptotic behavior of Kolmogorov entropy numbers of mixed derivative Sobolev spaces. Other intricate problems are, e.g., the dimension dependence of the star discrepancy of (classical) low-discrepancy constructions and the related conjecture of Woźniakowski [24, 40]. Powerful algorithms that approximate the star discrepancy could be used to make helpful numerical experiments and serve in this way, as Thiémard [47] put it, as "intuitive insight-providing tools."

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 [49] 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]. The L^2-star discrepancy has some well-known disadvantages (see [36]), but there are generalizations known which avoid those shortcomings; see [26, 45]. With the help of explicit formulas similar to that of Warnock, those generalized L^2-discrepancies can still be calculated within a reasonable time.

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, 47, 48]. 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

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

Then we obtain

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

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 [38, 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 [50], 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 an idea similar to that of the simulated annealing algorithm [33]. In [48], Thiémard gave an integer linear programming formulation for the problem and used techniques such 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 [42].

In this paper we present a new randomized algorithm to approximate the star discrepancy. Like 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 from the algorithm of Winker and Fang is a new nonuniform 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 $\lfloor r \rfloor := \max\{n \in \mathbb{Z} \,|\, n \le r\}$. For the purpose of readability we sometimes 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 $x \le y$ if $x_j \le y_j$ 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

$$\bar{\delta}(y) = \bar{\delta}(y, X) := \tfrac{1}{n}\bar{A}(y, X) - V_y \quad \text{and} \quad \delta^*(y) = \delta^*(y, X) := \max\big\{\delta(y), \bar{\delta}(y)\big\}.$$

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 [50] 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 [50].

Computation of threshold values. Let I be the number of iterations to be performed by the algorithm. Prior to the actual run of the algorithm, a sequence $(T(t))_{t=1}^I$ of nonpositive numbers satisfying $T(1) \le T(2) \le \cdots \le T(I)$ is computed (details are given below).

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

Optimization. In the tth iteration, the algorithm chooses a point x^{nb} uniformly at random from a given *neighborhood* $\mathcal{N}(x^c) \subseteq \bar{\Gamma}(X)$ of x^c and calculates $\delta^*(x^{nb})$. It then computes $\Delta\delta^* := \delta^*(x^{nb}) - \delta^*(x^c)$. If $\Delta\delta^* \ge T(t)$ for the nonpositive threshold value $T(t)$, then x^c is updated, i.e., the algorithm sets $x^c := x^{nb}$. With the help of the threshold $T(t)$ the algorithm is prevented from getting stuck in a bad local maximum x^c of δ^*—"local" with respect to the underlying neighborhood definition.

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

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

where $\phi_j : [|\bar{\Gamma}_j(X)|] \to \bar{\Gamma}_j(X)$ is the ordering of the set $\bar{\Gamma}_j(X)$, i.e., $\phi_j(r) < \phi_j(s)$ for $r < s$. That is, $\mathcal{N}_{k,j}(x)$ consists of x_j, the first ℓ coordinates in $\bar{\Gamma}_j(X)$ to the "left" of x_j, and the first ℓ coordinates in $\bar{\Gamma}_j(X)$ to the "right" of x_j. The *neighborhood* $\mathcal{N}_k^{j_1,\dots,j_{mc}}(x)$ *of* x *of order* k is the Cartesian product

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

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, $|\mathcal{N}_k^{j_1,\dots,j_{mc}}(x)| \le (2\ell+1)^{mc}$. 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. As mentioned above, the threshold values $(T(t))_{t=1}^{I}$ are chosen such that $T(1) \le T(2) \le \cdots \le T(I) \le 0$ holds. That is, they increase during the run of the algorithm. This is intended to enforce the algorithm to end up at a local maximum of δ^* which is reasonably close to $d_\infty^*(X)$.

The thresholds are chosen as follows. Let I again be the total number of iterations to be performed by the algorithm. Let $k \le n$ and $mc \le d$ be two variables. In a first step we compute $\sqrt{I}$ nonpositive real numbers as follows. For each $t \in [\sqrt{I}]$ we choose a grid point $y^t \in \bar{\Gamma}(X)$ uniformly at random, we then choose a neighbor $\tilde{y}^t \in \mathcal{N}_k^{mc}(y^t)$ uniformly at random, and we calculate the value $T'(t) := -|\delta^*(y^t) - \delta^*(\tilde{y}^t)|$. In the second step we sort the values $T'(1), \dots, T'(\sqrt{I})$ in increasing order. From this we obtain a sequence $(T''(t))_{t=1}^{\sqrt{I}}$ satisfying $T''(1) \le T''(2) \le \cdots \le T''(\sqrt{I}) \le 0$. For a given $\alpha \in (0.9, 1]$, the $\alpha\sqrt{I}$ values $T''(\sqrt{I} - \alpha\sqrt{I} + 1) \le \cdots \le T''(\sqrt{I})$ closest to zero are selected as the threshold sequence. Let $J = \alpha^{-1}\sqrt{I}$. This is the number of iterations performed for each threshold value. That is, for any $t \in [I]$ we set

$$T(t) := T''\left(\sqrt{I} - \alpha\sqrt{I} + \left\lceil \frac{t}{J} \right\rceil\right).$$

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 [51, 52]. 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.

First, this is not appropriate for higher-dimensional inputs: To illustrate this, let us consider random point sets of a fixed size n. With high probability their discrepancy will increase with the dimension d [25, 28]. Thus, if the expected discrepancy is c in some dimension, it is at least c in higher dimensions. In the case that the discrepancy of a set is caused by a half-open box that contains too few points, it is thus very likely that this box has volume at least c (otherwise the set would have a discrepancy strictly smaller than c); hence its average coordinate value should be at least $c^{1/d}$. This indicates that it is more appropriate for higher dimensional sets X to increase the weight of those points in the grid $\bar{\Gamma}(X)$ with larger coordinates and to decrease the weight of the points with small coordinates. (For a more rigorous analysis see section 5.)

Second, a uniform probability distribution does not take into account the topological characteristics of the point set X such as the 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 seems 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 nonuniform 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 nonuniform (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. The combination of these two steps makes it more likely that the grid point we end up with results in a test box with a sufficiently large volume and that it is the corner point of a grid cell in $\bar{\Gamma}(X)$ with relatively large volume. 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 nonuniform sampling strategy can be found in sections 5 and 6.

3.1. Sampling of neighbors. In the following, we present how we modify the probability distribution over the neighborhood sets. Our nonuniform 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 $\varphi_j : [|\bar{\Gamma}_j(X) \cup \{0\}|] \to \bar{\Gamma}_j(X) \cup \{0\}$ 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

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

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

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

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 nontrivial components, we choose a different probability distribution which we describe in the following. In this section, we shall only introduce the measure, but we provide some theoretical justification for this measure in section 5.1. First, let us consider the polynomial product measure

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

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 $y := (z_1^{1/d}, \ldots, z_d^{1/d})$.

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

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

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

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

If we want to sample a random point $y \in C_k^{j_1,\ldots,j_{mc}}(x)$, we randomly choose scalars $s_1, \ldots, s_{mc}$ in $[0,1]$ with respect to λ and put $y_{j_i} := \Psi_{j_i}^{-1}(s_i)$ for $i = 1, \ldots, mc$. For indices $j \notin \{j_1, \ldots, j_{mc}\}$ 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 $y_j^+ := \min\{x_j^i \in \bar{\Gamma}_j(X) \,|\, y_j \le x_j^i\}$. If $y_j \ge \min \bar{\Gamma}_j(X)$, we set $y_j^- := \max\{x_j^i \in \bar{\Gamma}_j(X) \,|\, y_j \ge x_j^i\}$, 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 $\delta(y^+) = V_{y^+} - A(y^+, X) \ge V_y - A(y, X) = \delta(y)$. Similarly, if $y_j \ge \min \Gamma_j(X)$ for all $j \in [d]$, we have $\bar{A}(y^-, X) = \bar{A}(y, X)$. Hence, $\bar{\delta}(y^-) = \bar{A}(y^-, X) - V_{y^-} \ge \bar{A}(y, X) - V_y = \bar{\delta}(y)$. If $y_j < \min \Gamma_j(X)$ for at least one $j \in [d]$, we have $\bar{\delta}(y) \le 0$ since $\bar{A}(y, X) = 0$. But we also have $A(y, X) = 0$, and thus $\delta^*(y) = \delta(y) \le \delta(y^+)$. Putting everything together, we have shown that $\max\{\delta(y^+), \bar{\delta}(y^-)\} \ge \delta^*(y)$.

Since it is of insignificant additional computational cost to also compute $\bar{\delta}(y^{-,-})$ where $y_j^{-,-} := y_j^-$ for all $j \in [d]$ with $y_j \ge \min \bar{\Gamma}_j(X)$ 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.

To sample a neighbor x^{nb} of x^c, the algorithm thus does the following. First, it samples mc coordinates $j_1, \ldots, j_{mc} \in [d]$ uniformly at random. Then it samples a point $y \in C_k^{j_1,\ldots,j_{mc}}(x^c)$ 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 $\Delta\delta^* = \delta_\Gamma^*(y) - \delta^*(x^c) \ge T$, 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. Since the starting point is sampled with respect to the nonuniform measure π^d, it is likely that most of the search steps in the actual algorithm will lead to results similar to the ones we obtain from our simulation to generate the threshold sequence.

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 a 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 only the points $y \in \bar{\Gamma}(X)$ and to calculate $\delta^*(y)$; cf. (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

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

We say that $S_j(y)$ is a $\delta(X)$-*critical surface* if $S_j(y) \cap \{x^1, \ldots, x^n\} \neq \emptyset$ 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.,

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

We say $\bar{S}_j(y)$ is a $\bar{\delta}(X)$-*critical surface* if $\bar{S}_j(y) \cap \{x_1, \ldots, x_n\} \neq \emptyset$. 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 $\phi_j : [\nu_j] \to \bar{\Gamma}_j(X)$ denote the ordering of $\bar{\Gamma}_j(X)$. Let $\Phi : [\nu_1] \times \cdots \times [\nu_d] \to \bar{\Gamma}(X)$ be the mapping with components ϕ_j, $j = 1, \ldots, d$. We say that a multi-index $(i_1, \ldots, i_d) \in [\nu_1] \times \cdots \times [\nu_d]$ is a $\delta(X)$-*critical multi-index* if $\Phi(i_1, \ldots, i_d)$ 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 nonempty 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. ∎

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 nontrivial 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 procedures.[1]

Snapping down. We aim at finding a $\bar{\delta}(X)$-critical point $y^{-,sn} \le y^-$ 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 $j \in [d]$

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

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 $x_j^i \le y_j^{-,sn}$ and update $y_j^{-,sn} := x_j^i$ otherwise.

Snapping up.[2] 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 $y^{+,sn} \ge y^+$ 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 upward. 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 $x \in \{x^i \,|\, i \in [n]\}$ we now do the following. If $x \in [0,y^+)$ or $x_j \ge y_j^{+,sn}$ 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 $x_{\sigma(j)} \ge y_{\sigma(j)}^+$. 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.

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

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

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 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 $y^{-,-,sn}$ 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 $t = 1, \ldots, \sqrt{I}$ we compute the tth 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 $T(t) := -|\delta^{*,sn}(y^t) - \delta^{*,sn}(\tilde{y}^t)|$, $t = 1, \ldots, \sqrt{I}$, in increasing order. This will be our threshold sequence.

4.2. Shrinking neighborhoods and a 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, toward 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 allowed to change only a 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 tth 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

tth 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 is obtained by an open, subproportionally 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, overproportionally 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 consider only 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 compute only y^+ (and $y^{+,sn}$), and we initialize $x^c := y^+$ (we update $x^c := y^+$ if and only if $\Delta\delta = \delta(y^{+,sn}) - \delta\big((x^c)^{+,sn}\big) \geq T$, 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 $\Delta\bar{\delta} = \max\{\bar{\delta}(y^{-,sn}), \bar{\delta}(y^{-,-,sn})\} - \bar{\delta}\big((x^c)^{-,sn}\big) \geq T$). 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 multicore 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. Of our main innovations, namely, the nonuniform sampling strategy and the rounding procedures "snapping up" and "snapping down," we already analyzed the snapping procedures and proved in Lemma 4.1 that they improve the quality of our estimates. The analysis of the nonuniform 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 nonuniform 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 $\bar{\Gamma}_j(X)$, $j = 1, \ldots, d$. 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 for choosing a probability measure on the neighborhoods that 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 nonuniform probability measure π^d leads to results superior to those obtained when sampling with respect to the Lebesgue measure λ^d.

Before we start with the theoretical analysis, let us give a strong indication that our nonuniform sampling strategy is much more appropriate in higher dimensions than a uniform sampling strategy. In [50] Winker and Fang chose in each of the dimensions $d = 4, 5, 6$ in a random manner ten 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 $\overline{\delta}$ is technically more involved than the proof of Proposition 5.1. 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 $\{r \in [0,1]^d \,|\, \delta(x^*) - \delta(r) \leq \varepsilon\}$, then*

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

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

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

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

then $\delta(x^*) - \delta^R \leq \varepsilon$ 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 $x \le x^*$ we get

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

Therefore, the Lebesgue measure of the set

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

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

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

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

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

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

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

and $x_j^{(k),i} < k/(k+1) - \varepsilon$ for all $i \in [n]$, $j > 1$. Then obviously $x^*(X^{(k)}) = (k/(k+1), 1, \ldots, 1)$, 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 $i \le R$. 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). ∎

For $d \ge 1$ and $\varepsilon \le 1/2$ we have $\big|\ln(1 - \varepsilon^d/d!)\big|^{-1} \sim d!\,\varepsilon^{-d}$. In this case we can only ensure 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 superexponential in d.

Let us end this section with some comments on the setting where we are interested only 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 $\{r \in [0,1]^d \,|\, \bar\delta(\bar x) - \bar\delta(r) \le \varepsilon\}$, there seems to be no alternative to estimating $\lambda^d(U(\bar x))$, where

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

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 to 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^* be 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 $\{r \in [0,1]^d \,|\, \delta(x^*) - \delta(r) \leq \varepsilon\}$, 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 $r^1, \ldots, r^R \in [0,1]^d$, sampled independently with respect to π^d, and put $\delta^R := \max_{i=1}^R \delta(r^i)$. If

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

then $\delta(x^*) - \delta^R \leq \varepsilon$ 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 $x_1^1 = \cdots = x_1^n = \varepsilon$ and $x_j^i < \varepsilon$ for all $i \in [n]$, $j > 1$. Furthermore, at least an ε-fraction of the points should be equal to $(\varepsilon, 0, \ldots, 0)$. Then obviously $x^*(X) = (\varepsilon, 1, \ldots, 1)$ 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 $(1-\varepsilon^d)^R \leq \epsilon$ if R satisfies (5.6). ∎

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 $\sup\{\delta(x) \,|\, x \in [0,1]^d\}$ is still exponential in d, but at least not superexponential 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 nonuniform 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 one-dimensional situation. Recall that in the one-dimensional case $\pi = \pi^1$ coincides with $\lambda = \lambda^1$.

To analyze the one-dimensional situation, let $X := (x^i)_{i=1}^n$ be the given point configuration in $[0,1)$. Without loss of generality we assume that $0 \leq x^1 < \cdots < x^n < 1$. Since $\delta^*(1) = 0$ we do not need to consider the whole grid $\bar{\Gamma}(X)$ but can restrict ourselves to the set $\Gamma(X) = \{x^1, \ldots, x^n\}$. 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, it is 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 $w_u(x^i)$, $i = 1, \ldots, n$, 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 τ, $\bar{\tau} \in \Gamma(X)$ 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*

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

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 $\Gamma(X) := \{x^1, \ldots, x^{n+1}\}$. 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

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

and

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

we get

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

On the other hand, we have

$$\mathbb{E}(\delta) = \sum_{i=1}^{n+1} \frac{1}{n+1}\delta(x^i) = \frac{\delta(x^\nu)}{n+1} + \sum_{i=1}^{n} \frac{1}{n+1}\Big(\tilde{x}^i - \frac{i-1}{n+1}\Big)$$
$$= \frac{\delta(x^\nu)}{n+1} + \Big(\frac{n}{n+1}\Big)^2 \sum_{i=1}^{n} \frac{1}{n}\Big(\hat{x}^i - \frac{i-1}{n}\Big).$$

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. ∎

5.2.2. Analysis of higher dimensional situations $d \geq 2$. In dimension $d \geq 2$ a rigorous analysis is much harder than in the one-dimensional case, especially if we want to take into account the snapping procedures. A direct generalization of Lemma 5.3 is not possible, since one can easily construct point configurations in dimension $d = 2$ which do not satisfy the weight inequalities (5.7). We give some examples in [21], but one has to keep in mind that these point configurations are rather artificial and not well distributed at all. We believe that in generic cases or for well-distributed point sets our probability distributions on $\bar{\Gamma}(X)$ and $\Gamma(X)$ are in general superior to the uniform distribution on these grids. Further theoretical results would be interesting.

5.3. The chances of a grid point being a critical point. If $X = (x^i)_{i=1}^n$ is a sequence in $[0,1)^d$, which has been chosen randomly with respect to the Lebesgue measure, and if $x \in \bar{\Gamma}(X)$, then the larger the components of x are, the higher the probability of x being a $\delta(X)$-critical point is. The same holds for $\tilde{x} \in \Gamma(X)$ and $\bar{\delta}(X)$, respectively.

PROPOSITION 5.4. *Consider $[0,1)^{nd}$ as a probability space endowed with the probability measure λ^{nd}. Let $\iota := (i_1, \dots, i_d) \in [n+1]^d$. If k indices $i_{\nu(1)}, \dots, i_{\nu(k)}$ of the i_j, $j = 1, \dots, d$, are at most n and the remaining $d-k$ of them are equal to $n+1$, then for uniformly distributed random variable X in $[0,1)^{nd}$ the multi-index ι is $\delta(X)$-critical with probability*

$$\Big(\frac{(n-k)!}{n!}\Big)^{k-1} \prod_{j=1}^{k} \Big(\prod_{\ell=1}^{k-1} \max\{i_{\nu(j)} - \ell, 0\} \Big).$$

Proof. Let $\Phi = (\phi_1, \dots, \phi_d)$ be as in section 4.1. Since the event that for all coordinates $j \in [d]$ we have $|\bar{\Gamma}_j(X)| = n+1$ holds with probability 1, we restrict ourselves to this situation.

Without loss of generality, we may assume that $i_1, \dots, i_k \leq n$ and $i_{k+1} = \cdots = i_d = n+1$. Obviously, $S_{k+1}(\Phi(\iota)), \dots, S_d(\Phi(\iota))$ are δ-critical surfaces, since $\phi_{k+1}(n+1) = \cdots = \phi_d(n+1) = 1$. For $i = 1, \dots, k$ let $\sigma_i = \sigma_i(X) : [n] \to [n]$ be the permutation with

$$x_i^{\sigma_i(1)} < x_i^{\sigma_i(2)} < \cdots < x_i^{\sigma_i(n)} < 1.$$

Clearly, $\Phi(\iota)_j = \phi_j(i_j) = x_j^{\sigma_j(i_j)}$ for all $j \in [k]$. Since $S_i(x) \cap S_j(x) = \emptyset$ for all $i \neq j$ and all $x \in [0,1]^d$, the surfaces $S_1(\Phi(\iota)), \dots, S_k(\Phi(\iota))$ can only be $\delta(X)$-critical if $|\{\sigma_1(i_1), \dots, \sigma_k(i_k)\}| = k$. More precisely, ι is a $\delta(X)$-critical multi-index if and only if the condition

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

holds. This is equivalent to the k conditions

$$\begin{array}{lllll} \sigma_1^{-1}(\sigma_2(i_2))\,,\, \sigma_1^{-1}(\sigma_3(i_3))\,,\ldots\,, & \sigma_1^{-1}(\sigma_k(i_k)) & < & i_1\,, \\ \sigma_2^{-1}(\sigma_1(i_1))\,,\, \sigma_2^{-1}(\sigma_3(i_3))\,,\ldots\,, & \sigma_2^{-1}(\sigma_k(i_k)) & < & i_2\,, \\ \vdots \qquad\qquad \vdots & \vdots & \vdots & \vdots \\ \sigma_k^{-1}(\sigma_1(i_1))\,,\, \sigma_k^{-1}(\sigma_2(i_2))\,,\ldots\,, & \sigma_k^{-1}(\sigma_{k-1}(i_{k-1})) & < & i_k\,. \end{array} \tag{5.8}$$

Since all the components x_j^i, $i \in [n]$, $j \in [d]$, of X are independent random variables, we have that for a fixed index $\nu \in [d]$ each permutation $\tau : [n] \to [n]$ is equally likely to fulfill $\sigma_\nu(X) = \tau$. Thus, the probability of ι being a $\delta(X)$-critical index is just the number of k-tuples $(\sigma_1, \ldots, \sigma_k)$ of permutations fulfilling (5.8), divided by $(n!)^k$.

For given pairwise distinct values $\sigma_1(i_1), \ldots, \sigma_k(i_k)$ the jth condition of (5.8) is satisfied by $((i_j - 1) \ldots (i_j - (k-1)))(n-k)!$ 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. ∎

To state the corresponding proposition for $\bar{\delta}$, we have to introduce *Stirling numbers of second kind* $S(d,k)$. For $k \in \mathbb{N}$, $k \le d$, let $S(d,k)$ denote the number of partitions of $[d]$ into k nonempty 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 be found in, e.g., [41].

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

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

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

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

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

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

is satisfied. This can be rewritten as

$$\begin{array}{lllll} \sigma_1^{-1}(\sigma_2(i_2))\,,\, \sigma_1^{-1}(\sigma_3(i_3))\,,\ldots\,, & \sigma_1^{-1}(\sigma_d(i_d)) & \le & i_1\,, \\ \sigma_2^{-1}(\sigma_1(i_1))\,,\, \sigma_2^{-1}(\sigma_3(i_3))\,,\ldots\,, & \sigma_2^{-1}(\sigma_d(i_d)) & \le & i_2\,, \\ \vdots \qquad\qquad \vdots & \vdots & \vdots & \vdots \\ \sigma_d^{-1}(\sigma_1(i_1))\,,\, \sigma_d^{-1}(\sigma_2(i_2))\,,\ldots\,, & \sigma_d^{-1}(\sigma_{d-1}(i_{d-1})) & \le & i_d\,. \end{array} \tag{5.9}$$

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

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

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. ∎

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 [50], and also give a brief comparison against the genetic algorithm of Shah [42] and the integer programming-based algorithm of Thiémard [48].

6.1. Experimental setup. We divide our experiments into a thorough comparison against the algorithm of Winker and Fang [50], given in section 6.3, and more brief comparisons against the algorithms of Shah [42] and Thiémard [48], in sections 6.4 and 6.5, respectively. The algorithms `TA_basic` and `TA_improved`, as well as the algorithm of Winker and Fang [50], were implemented by the authors in the C programming language, based on the code used in [51, 52]. All implementations were done with equal care. In the case of Winker and Fang [50], 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 Thiémard [48], E. Thiémard 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 Thiémard was written for an older version. Finally, Shah has provided us with the application used in the experiments of [42], 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 [42]. Random numbers were generated using the Gnu C library pseudorandom 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 homepage,[3] or via the integer programming-based algorithm of Thiémard [48]. 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 neighborhood 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 [51, 52]. 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 we call this the *best-of-10 value*. This estimation is done via a bootstrapping method, as suggested by Johnson [32]; specifically, we perform 100 independent trials and use the outcomes of these trials as a sample space, computing (analytically) the expectation of a best-of-10 value sampled from this smaller space. The result is found to be stable over

[3] http://www.mpi-inf.mpg.de/∼wahl/.

re-executions. The comparisons are based on a fixed 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 [50] has the advantage that all algorithms compared use the same code base, compiler, and libraries, including the choice of pseudorandom number generator. This further removes differences that are not interesting to us.

6.2. Instances. Our point sets are of four types: Halton sequences [22], Faure sequences [15], Sobol' point sets [46], and so-called good lattice points (GLPs) [50]. The Halton sequences and GLPs were generated by programs written by the authors, the Faure sequences by a program of Burkardt [4], and the Sobol' sequences using the data and code of Joe and Kuo [31, 34].

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 nonexistent.

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 [50]. 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 [50] 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 note only that our re-implementation of the Winker and Fang algorithm gives notably worse results than what was reported in [50] for the same instances, in a way that cannot be fully explained by parameter settings. A comparison of the mean values of 100 runs of 10,000 iterations in dimensions $4, 5$, and 6 provided by the original implementation of Winker and Fang and by our re-implementation shows that our implementation gives results that are on average 2.1%

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.*

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

smaller than the numbers originally reported in [50, Table 2]. Except for a few instances, the differences between the results of both implementations are less than 6%.[4]

After extensive experimentation, we have no viable explanation for this difference. One candidate would be the pseudorandom number generator used, as [50] uses a random number library that we do not have access to. However, experiments with different random number generators (specifically, the 11 different standard generators available in the GNU Scientific Library) have not been found to make any significant difference. Still, our algorithms, and `TA_improved` in particular, perform well compared to the results reported in [50] (again, see [21] for the comparison).

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

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.*

			$d^*_\infty(\cdot)$	`TA_basic`		`TA_improved`		Winker & Fang	
Name	d	n	found	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

[4]The numbers underlying this comparison can be found in [51, page 74].

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

The highest-dimensional sets ($d = 50$) illustrate the deterioration of the Winker and Fang algorithm 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).

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* [42].

				`TA_basic`		`TA_improved`		Shah	
Class	d	n	$d^*_\infty(\cdot)$	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	$0^{(1)}$	$0.0689^{(1)}$

6.4. Comparisons with the algorithm by Shah. Table 6.3 lists the point sets used by Shah [42]. 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 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 of our algorithms. We also find (again) that `TA_improved` has 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 [48]. Since [48] 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, [48] 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 [47]; this is freely downloadable from Thiémard's homepage. Its prime feature is that it provides upper bounds with a nontrivial running time guarantee. The lower bounds that it produces are not as helpful as the upper bounds; e.g., it

TABLE 6.4

Comparison against the integer programming-based algorithm of Thiémard [48]. *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* [48] *to return a first output, and the value of this output; the last two columns report the lower bound reached by* [48] *if allocated the same time that* `TA_improved` *needs for completion, and the time required by* [48] *to match the result of* `TA_improved`.

	`TA_improved`		Thiémard: Initial		Same time,	Same result,
Instance	Time	Result	Time	Result	result	time
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

was reported in [11] and [42] that the lower bounds from the preliminary version of `TA_basic` [51, 52] and the genetic algorithm of Shah [42] 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 improve the quality of the lower bounds, in particular for the difficult higher-dimensional problem instances. Also the comparison with the algorithms by Thiémard 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 dimensions.

Appendix. 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*

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

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

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

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,

$$\begin{aligned}\lambda^d(A_\varepsilon(z)) &= \int_{\alpha_1}^{z_1}\int_{\tilde\alpha_1}^{\tilde z_1}\cdots\int_{\tilde\alpha_{d-1}}^{\tilde z_{d-1}} d\zeta_d\ldots d\zeta_2\, d\zeta_1\\ &= \int_{\alpha_1}^{z_1}\left(V_{\tilde z}-(V_{\tilde z}-\tilde\varepsilon)\sum_{k=0}^{d-2}\frac{(-\ln(1-\tilde\varepsilon/V_{\tilde z}))^k}{k!}\right)d\zeta_1\\ &= V_z-(V_z-\varepsilon)-(V_z-\varepsilon)\left[\sum_{k=1}^{d-1}\frac{1}{k!}\ln\Big(\frac{V_{\tilde z}}{V_z-\varepsilon}\zeta_1\Big)^k\right]_{\zeta_1=\alpha_1}^{z_1}\\ &= V_z-(V_z-\varepsilon)\sum_{k=0}^{d-1}\frac{(-\ln(1-\varepsilon/V_z))^k}{k!}\,. \qquad \square\end{aligned}$$

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

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

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

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

The power series converges for each $\epsilon>0$ uniformly and absolutely for all $V_z\in[\varepsilon+\epsilon,1]$. 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 $\lambda^d(A_\varepsilon(z))=V_zR(\varepsilon/V_z,d)$. Since $\partial_xR(x,d)=(-\ln(1-x))^{d-1}/(d-1)!$, it is sufficient to prove the following statement by induction over d:

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

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 $b_1(d) = d(d-1)/2(d+1)$. ∎

We now consider the polynomial product measure π^d.

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

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

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

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

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

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 $0 \leq r \leq \varepsilon$

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

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:

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

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. ∎

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. ∎

Acknowledgments. We gratefully acknowledge Manan Shah, Eric Thiémard, and Peter Winker for providing us with source code and implementations of the applications used in their experiments (in [42], [48], and [50], respectively) and in general for their helpful comments. We thank two anonymous referees for useful comments which helped to improve the presentation of our results.

REFERENCES

[1] I. ALTHÖFER AND K.-U. KOSCHNICK, *On the convergence of "threshold accepting,"* Appl. Math. Optim., 24 (1991), pp. 183–195.

[2] J. BECK AND W. W. L. CHEN, *Irregularities of Distribution*, Cambridge University Press, Cambridge, UK, 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, UK, 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, UK, 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., Springer, Berlin, Heidelberg, 2009, pp. 323–338.

[11] B. DOERR, M. GNEWUCH, AND M. WAHLSTRÖM, *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*, Lecture Notes in Math. 1651, Springer, Berlin, 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. Giannopoulos, C. Knauer, M. Wahlström, and D. Werner, *Hardness of discrepancy computation and epsilon-net verification in high dimension*, J. Complexity, 28 (2012), pp. 162–176.
[18] M. Gnewuch, *Bracketing numbers for axis-parallel boxes and applications to geometric discrepancy*, J. Complexity, 24 (2008), pp. 154–172.
[19] M. Gnewuch, *Weighted geometric discrepancies and numerical integration on reproducing kernel Hilbert spaces*, J. Complexity, 28 (2012), pp. 2–17.
[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*, extended preprint, http://arXiv.org/abs/1103.2102v2, 2011.
[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] S. Heinrich, *Some open problems concerning the star-discrepancy*, J. Complexity, 19 (2003), pp. 416–419.
[25] S. Heinrich, E. Novak, G. W. Wasilkowski, and H. Woźniakowski, *The inverse of the star-discrepancy depends linearly on the dimension*, Acta Arith., 96 (2001), pp. 279–302.
[26] F. J. Hickernell, *A generalized discrepancy and quadrature error bound*, Math. Comp., 67 (1998), pp. 299–322.
[27] F. J. Hickernell, I. H. Sloan, and G. W. Wasilkowski, *On tractability of weighted integration over bounded and unbounded regions in $\mathbb{R}^s$*, Math. Comp., 73 (2004), pp. 1885–1901.
[28] A. Hinrichs, *Covering numbers, Vapnik-Červonenkis classes and bounds for the star-discrepancy*, J. Complexity, 20 (2004), pp. 477–483.
[29] A. Hinrichs, F. Pillichshammer, and W. Ch. Schmid, *Tractability properties of the weighted star discrepancy*, J. Complexity, 24 (2008), pp. 134–143.
[30] 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., Springer, Berlin, 2006, pp. 181–196.
[31] S. Joe and F. Y. Kuo, *Constructing Sobol' sequences with better two-dimensional projections*, SIAM J. Sci. Comput., 30 (2008), pp. 2635–2654.
[32] 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., AMS, Providence, RI, 2002, pp. 215–250.
[33] S. Kirckpatrick, C. Gelatt, and M. Vecchi, *Optimization by simulated annealing*, Science, 20 (1983), pp. 671–680.
[34] F. Y. Kuo, *Sobol Sequence Generator*, http://web.maths.unsw.edu.au/~fkuo/sobol/index.html, 2010.
[35] C. Lemieux, *Monte Carlo and Quasi-Monte Carlo Sampling*, Springer, New York, 2009.
[36] J. Matoušek, *On the L_2-discrepancy for anchored boxes*, J. Complexity, 14 (1998), pp. 527–556.
[37] J. Matoušek, *Geometric Discrepancy*, 2nd ed., Springer, Berlin, 2010.
[38] H. Niederreiter, *Discrepancy and convex programming*, Ann. Mat. Pura Appl., 93 (1972), pp. 89–97.
[39] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 63, SIAM, Philadelphia, 1992.
[40] E. Novak and H. Woźniakowski, *Tractability of Multivariate Problems. Vol. 2: Standard Information for Functionals*, EMS Tracts Math., European Mathematical Society (EMS), Zürich, 2010.
[41] J. Riordan, *An Introduction to Combinatorial Analysis*, Wiley, New York, 1958.
[42] M. Shah, *A genetic algorithm approach to estimate lower bounds of the star discrepancy*, Monte Carlo Methods Appl., 16 (2010), pp. 379–398.
[43] V. Sinescu and S. Joe, *Good lattice rules based on the general weighted star discrepancy*, Math. Comp., 76 (2007), pp. 989–1004.

[44] 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, http://mcqmc.mimuw.edu.pl/?page=presentations, 2010.

[45] I. H. Sloan and H. Woźniakowski, *When are quasi-Monte Carlo algorithms efficient for high dimensional integrals?*, J. Complexity, 14 (1998), pp. 1–33.

[46] I. M. Sobol', *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).

[47] E. Thiémard, *An algorithm to compute bounds for the star discrepancy*, J. Complexity, 17 (2001), pp. 850–880.

[48] E. Thiémard, *Optimal volume subintervals with k points and star discrepancy via integer programming*, Math. Methods Oper. Res., 54 (2001), pp. 21–45.

[49] T. T. Warnock, *Computational investigations of low-discrepancy point sets*, in Applications of Number Theory to Numerical Analysis, S. K. Zaremba, ed., Academic Press, New York, 1972, pp. 319–343.

[50] P. Winker and K.-T. Fang, *Application of threshold-accepting to the evaluation of the discrepancy of a set of points*, SIAM J. Numer. Anal., 34 (1997), pp. 2028–2042.

[51] C. Winzen, *Approximative Berechnung der Sterndiskrepanz*, Diplomarbeit, Mathematisches Seminar, Christian-Albrechts-Universität zu Kiel, Kiel, available online from http://www.mpi-inf.mpg.de/∼winzen/WinzenDiplomarbeit.pdf, 2007.

[52] C. Winzen, *Approximative Berechnung der Sterndiskrepanz*, VDM Dr. Müller, Saarbrücken, Germany, 2010.