Implementation of a Component-By-Component Algorithm to Generate Small Low-Discrepancy Samples

Benjamin Doerr, Michael Gnewuch, and Magnus Wahlström

Abstract In [B. Doerr, M. Gnewuch, P. Kritzer, F. Pillichshammer. Monte Carlo Methods Appl., 14:129–149, 2008], a component-by-component (CBC) approach to generate small low-discrepancy samples was proposed and analyzed. The method is based on randomized rounding satisfying hard constraints and its derandomization. In this paper we discuss how to implement the algorithm and present first numerical experiments. We observe that the generated points in many cases have a significantly better star discrepancy than what is guaranteed by the theoretical upper bound. Moreover, we exhibit that the actual discrepancy is mainly caused by the underlying grid structure, whereas the rounding errors have a negligible contribution. Hence to improve the algorithm, we propose and analyze a randomized point placement. We also study a hybrid approach which combines classical low-discrepancy sequences and the CBC algorithm.

1 Introduction

The *star discrepancy* of an N-point set $\mathscr{P}_s = \{p_0, \ldots, p_{N-1}\}$ in the s-dimensional unit cube $[0,1]^s$ is defined by

$$D_N^*(\mathscr{P}_s) := \sup_{x \in [0,1]^s} |\Delta_s(x, \mathscr{P}_s)|.$$

Here the discrepancy function Δ_s of the set $\mathscr{P}_s$ is given, for $x = (x_1, \ldots, x_s)$, by

Benjamin Doerr and Magnus Wahlström
Max-Planck-Institut für Informatik, Saarbrücken, Germany
http://www.mpi-inf.mpg.de/~{doerr|wahl}

Michael Gnewuch
Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Kiel, Germany
http://www.numerik.uni-kiel.de/~mig/

$$\Delta_s(x,\mathscr{P}_s) = \lambda_s([0,x)) - \frac{1}{N}\sum_{j=0}^{N-1} 1_{[0,x)}(p_j),$$

where λ_s is the s-dimensional Lebesgue measure and $1_{[0,x)}$ is the characteristic function of the s-dimensional half-open box $[0,x) = [0,x_1)\times\cdots\times[0,x_s)$.

It is well known that the star discrepancy is intimately related to multivariate integration. If we have a function f defined on $[0,1]^s$ and a point set $\mathscr{P}_s = \{p_0,\ldots,p_{N-1}\}$, then the Koksma-Hlawka inequality states

$$\left|\int_{[0,1]^s} f(x)\,\mathrm{d}x - \frac{1}{N}\sum_{k=0}^{N-1} f(p_k)\right| \le D_N^*(\mathscr{P}_s)V(f),$$

where $V(f)$ denotes the variation of f in the sense of Hardy and Krause, see, e.g., [11, 12]. Thus for multivariate integration it is important to find small point sets with low discrepancy.

For fixed dimension s various N-point sets $\mathscr{P}_s$ have been constructed satisfying

$$D_N^*(\mathscr{P}_s) \le C_{\mathscr{P}_s}\log(N)^{s-1}N^{-1}, \quad N \ge 2, \tag{1}$$

with $C_{\mathscr{P}_s}$ a suitable constant depending on $\mathscr{P}_s$, see, e.g., [12]. These constructions typically suffer from two difficulties. One is that often the constant $C_{\mathscr{P}_s}$ is not known sufficiently well or depends unfavorably on the dimension s. The other is that these bounds for large s and moderate N unfortunately give no useful information, since $\log(N)^{s-1}N^{-1}$ is an increasing function in N for $N \le e^{s-1}$.

A bound more helpful for high-dimensional integration was established by Heinrich et al. [10]. They proved in a non-constructive way that there is a constant $c > 0$ such that for all $s,N \in \mathbb{N}$ an N-point set $\mathscr{P}_s \subseteq [0,1)^s$ exists satisfying

$$D_N^*(\mathscr{P}_s) \le cs^{1/2}N^{-1/2}. \tag{2}$$

Which point sets do satisfy bounds like (2)? Discrepancy calculations performed by Thiémard [17, 18] indicate that known constructions which exhibit the asymptotic behavior (1) may not satisfy (2). As pointed out in [9], actually the seemingly easier question if any of the known constructions of low-discrepancy point sets $\mathscr{P}_{N,s}$ satisfies estimates of the form $D_N^*(\mathscr{P}_{N,s}) \le cs^\kappa N^{-\alpha}$ for all $s,N \in \mathbb{N}$, where c,κ,α are positive constants not depending on N and s, still remains open.

In [1, 3] the first non-trivial deterministic algorithmic point constructions were proposed whose star discrepancy grows for a fixed number of points with respect to the dimension s at most like $\sqrt{s}$. A drawback of these algorithms is their large runtime. A considerable speed-up was achieved by Kritzer, Pillichshammer and the first two authors [2]. They perform the derandomization in a component-by-component (CBC) fashion. An additional advantage of this new algorithm is that it allows a simple exact computation of the star discrepancy of the output point set.

The disadvantage is that the theoretical upper bound on the star discrepancy of the generated points grows like $s^{3/2}$ (if one generates the set component by com-

ponent starting in dimension 1) instead of like $\sqrt{s}$. The paper gives no indication of whether this is likely to happen in practice or not. A second question left open in [2] is how useful it is in practice to extend given low-discrepancy point sets in the dimension via the CBC algorithm. This approach is related to padding quasi-Monte Carlo (QMC) by Monte Carlo (MC), see also Sect. 3.5.

For these reasons, we implemented the CBC algorithm in the programming language C99. In this paper, we describe some details of the implementation and report the results of several numerical experiments. Among other outcomes, they show that the discrepancy of the point sets constructed by our algorithm for reasonable settings grows only linearly in the dimension s. Also, we do observe a significant advantage of extending existing good point sets in small dimensions via our approach to higher dimensions.

Another motivation for this paper from a different subarea of discrepancy theory is the following. There is a vast literature on the theory of randomized rounding and its derandomization, starting with the monograph of Spencer [16] and the celebrated paper of Raghavan [14]. But although derandomization is an algorithmic tool, seemingly none of these publications cares about practical aspects as, e.g., explicit descriptions of the resulting algorithms, the run-times of algorithms, or the rounding errors observed in practice. It seems that this work, together with [4] are the first to fight this short-coming.

2 Description of the Algorithm

2.1 General Method

The CBC approach presented in [2] aims at using an existing $(s-1)$-dimensional low-discrepancy point set to construct an s-dimensional one. Let a point set $\mathscr{P}_{s-1} = \{y_0,\dots,y_{N-1}\} \subset [0,1)^{s-1}$ be given. For an integer $m_s \geq 2$ the algorithm determines numbers $X_0,\dots,X_{N-1}$, each chosen from the set

$$G_s = \left\{ \frac{1}{2m_s}, \frac{3}{2m_s}, \dots, \frac{2m_s-1}{2m_s} \right\} \tag{3}$$

and returns a point set $\mathscr{P}_s = \mathscr{P}_s(X_0,\dots,X_{N-1}) := \{(y_0,X_0),\dots,(y_{N-1},X_{N-1})\} \subset [0,1)^s$. If we choose

$$m_s = m_s(N) := \left\lceil \frac{\sqrt{N}}{\sqrt{2}} \left(s \log\left(\rho'(N,s)\right) + \log(4) \right)^{-1/2} \right\rceil, \tag{4}$$

where

$$\rho'(N,s) = 2\sqrt{\mathrm{e}} \left(\max\{1, N/((1+2\log(2))s)\}\right)^{1/2},$$

then the output set $\mathscr{P}_s$ satisfies the discrepancy bound

$$D_N^*(\mathscr{P}_s) \le \left(\sqrt{3}+\frac{1}{\sqrt{2}}\right)\frac{\sqrt{s}}{\sqrt{N}}\left(\log\left(\rho'(N,s)\right)+\frac{1}{s}\log(4)\right)^{1/2}+D_N^*(\mathscr{P}_{s-1}). \quad (5)$$

Here we confine ourselves to sets $\mathscr{P}_s$ that are subsets of the anisotropic grid

$$\mathscr{G}_s := G_1 \times \cdots \times G_s,$$

where G_d is defined as in (3) and (4), but with s replaced by d for $d = 1,\ldots,s$. Let

$$G_d^* := \left\{\frac{1}{m_d},\frac{2}{m_d},\ldots,\frac{m_d-1}{m_d},1\right\}$$

for $d = 1,\ldots,s$, and let

$$\mathscr{G}_s^* := G_1^* \times \cdots \times G_s^*.$$

Let $t_1,\ldots,t_n$, $n := \Pi_{d=1}^s m_d$, be an enumeration of $\mathscr{G}_s^*$ and

$$\mathscr{T}_s^* := \{[0,t_i) \,|\, i = 1,\ldots,n\}.$$

When deciding the values for $X_0,\ldots,X_{N-1} \in G_s$, our algorithm tries to obtain a low discrepancy in all boxes of $\mathscr{T}_s^*$. As described in [2], this together with the fact that $\mathscr{P}_s \subset \mathscr{G}_s$ allows to calculate the exact star discrepancy of the output set within the course of the algorithm without essentially increasing the effort.

Formulation as Rounding Problem

The CBC algorithm derandomizes randomized rounding satisfying hard constraints. To describe it more precisely, let us formulate our problem as a rounding problem.

As discussed before, if $\mathscr{P}_{s-1} = \{y_0,\ldots,y_{N-1}\} \subset \mathscr{G}_{s-1}$ is given, we aim at finding $X_0,\ldots,X_{N-1} \in G_s$ such that $\mathscr{P}_s = \{(y_j,X_j) : 0 \le j < N\}$ has small discrepancy in the sense of (5).

Let $\mathscr{X}$ be the set of all $x \in [0,1]^{\{0,\ldots,N-1\}\times\{1,\ldots m_s\}}$ such that $x\mathbf{1}_{m_s} = \mathbf{1}_N$. Let $\bar{x} \in \mathscr{X}$ be defined by $\bar{x}_{jk} = \frac{1}{m_s}$ for all j,k. A *rounding* of $\bar{x}$ is an $x \in \mathscr{X} \cap \{0,1\}^{\{0,\ldots,N-1\}\times\{1,\ldots m_s\}}$. That is, each x_{jk} is a rounding of $\bar{x}_{jk}$ and the *hard constraints* $\sum_{k=1}^{m_s} x_{jk} = 1$ for all j are satisfied.

Let us define the linear function $A : \mathbb{R}^{\{0,\ldots,N-1\}\times\{1,\ldots m_s\}} \to \mathbb{R}^n$ by

$$A(x)_i = \sum_{j=0}^{N-1}\sum_{k=1}^{m_s} x_{jk}1_{[0,t_i)}(y_j,\hat{k}) \quad \text{for } i = 1,\ldots,n;$$

here we used the shorthand $\hat{k} = \frac{2k-1}{2m_s}$ for the kth point of G_s. If $x \in \mathscr{X}$ is binary, put $X_j = \hat{k}_j$ for $j = 0,1,\ldots,N-1$, where k_j is the index for which $x_{jk_j} = 1$. Then

$$\mathscr{P}(x) := \mathscr{P}(X_0,\ldots,X_{N-1}) = \{(y_j,X_j) : 0 \le j < N\}$$

is an N–point set in $\mathscr{G}_s \subset [0,1]^s$ and $A(x)_i$ equals $|\mathscr{P}(x) \cap [0,t_i)|$. In [2, Sect. 4] an elementary calculation showed that

$$D_N^*(\mathscr{P}(x)) \leq \tfrac{1}{N}\|A(x)-A(\bar{x})\|_\infty + D_N^*(\mathscr{P}_{s-1}) + \tfrac{1}{2m_s}. \tag{6}$$

This implies that point sets $\mathscr{P}(x)$ with small discrepancy correspond to roundings $x \in \mathscr{X}$ of $\bar{x}$ with small rounding error $\|A(x)-A(\bar{x})\|_\infty$.

Solving the Rounding Problem

How does our algorithm generate roundings x that satisfy the hard constraints $\sum_{k=1}^{m_s} x_{jk} = 1$ for all j and exhibit a small rounding error? The randomized construction is to choose for each j independently a $k_j \in \{1,\ldots,m_s\}$ at random such that $\Pr[k_j = k] = \bar{x}_{jk}$ for all j,k. Then for all j we define binary random variables X_{jk} by $X_{jk} = 1$ if and only if $k = k_j$, and $X_{jk} = 0$ otherwise. Note that any outcome of X lies in $\mathscr{X}$. Put

$$\sigma := \frac{\sqrt{N}}{\sqrt{2}}\left(s\log(\rho'(N,s)) + \log(4)\right)^{1/2}. \tag{7}$$

As shown in [2], we get $P := \sum_i \Pr[|(A(X-\bar{x}))_i| \geq \sigma] \leq 1/2$ ("small initial failure probability"). In particular, there is an $x \in \mathscr{X}$ such that $|(A(x-\bar{x}))_i| \leq \sigma$ for all i.

We can compute such roundings x by derandomizing the probabilistic construction above. For $k = 1,\ldots,m_s$, let us consider the conditional probability

$$P_k := \sum_i \Pr\left[|(A(X-\bar{x}))_i| \geq \sigma \,\middle|\, k_0 = k\right].$$

Since $P = \sum_{k=1}^{m_s} \frac{1}{m_s} P_k$, there is a $1 \leq k_0^* \leq m_s$ such that $P_{k_0^*} \leq P \leq 1/2$ ("decreasing failure probability"). Next, let

$$P_{k_0^* k} := \sum_i \Pr\left[|(A(X-\bar{x})_i| \geq \sigma \,\middle|\, k_0 = k_0^*, k_1 = k\right].$$

Again, $P_{k_0^*} = \sum_{k=1}^{m_s} \frac{1}{m_s} P_{k_0^* k}$, and there is a $1 \leq k_1^* \leq m_s$ such that $P_{k_0^* k_1^*} \leq P_{k_0^*} \leq 1/2$. Proceeding like this we end up with $k_0^*,\ldots,k_{N-1}^*$ such that

$$P_{k_0^*\ldots k_{N-1}^*} := \Pr\left[|(A(X-\bar{x}))_i| \geq \sigma \,\middle|\, \forall\, 0 \leq j < N : k_j = k_j^*\right] \leq 1/2.$$

Since $P_{k_0^*\ldots k_{N-1}^*}$ involves no randomness, we actually have $P_{k_0^*\ldots k_{N-1}^*} = 0$. Then we define x as follows: For each $0 \leq j < N$, we set $x_{jk_j^*} := 1$ and $x_{jk} := 0$ for all other k. This yields a binary $x \in \mathscr{X}$ such that $|(A(x-\bar{x}))_i| \leq \sigma$ for all i.

In practice we usually cannot compute the conditional probabilities $P_{k_0^* k_1^* \ldots}$ in time polynomially bounded in N, m_s and n. However, we can compute (in polynomial time) upper bounds $U_{k_0^* k_1^* \ldots}$ for the exact conditional probabilities $P_{k_0^* k_1^* \ldots}$ such that the following key properties are maintained:

- Small initial (estimated) failure probability: $U < 1$.
- Decreasing (estimated) failure probability: For all $0 \le \ell < N$ and $k_0^*, \dots, k_{\ell-1}^* \in \{1, \dots, m_s\}$ there is a $1 \le k \le m_s$ such that $U_{k_0^* k_1^* \dots k_{\ell-1}^* k} \le U_{k_0^* k_1^* \dots k_{\ell-1}^*}$.

The quantities $U_{k_0^* k_1^* \dots}$ are called *pessimistic estimators* for the conditional probabilities $P_{k_0^* k_1^* \dots}$. This notion was introduced by Raghavan [14], who also showed that such pessimistic estimators exist for the conditional probabilities that occur in our derandomization.

To achieve that the initial estimated failure probability U is less than one, we have to choose σ in equation (7) to be $\sqrt{6}$ times larger than there. This choice implies that the output set $\mathscr{P}_s$ of the CBC algorithm satisfies the discrepancy bound (5).

With suitable implementation, the run-time of the derandomized algorithm is $O(nNm_s)$. Under the assumption $s \le N/3$, this is

$$O(nNm_s) = O\left(\frac{c^s N^{\frac{s+3}{2}}}{s^{\frac{s}{2}+\frac{3}{4}} \log\left(\frac{N}{s}\right)^{\frac{s+1}{2}}}\right),$$

where c is some constant independent of N and s.

We can use the derandomized algorithm iteratively. That is, we first use it to generate a point set $\mathscr{P}_1 \subset [0,1)$ and then to repeatedly add dimensions until we obtain the desired point set $\mathscr{P}_s$. When doing so, the final point set $\mathscr{P}_s$ satisfies the discrepancy estimate

$$D_N^*(\mathscr{P}_s) \le \left(\sqrt{3} + \frac{1}{\sqrt{2}}\right) \frac{s^{3/2}}{\sqrt{N}} \left(\log\left(\rho'(N,s)\right) + \frac{1}{s}\log(4)\right)^{1/2}.$$

2.2 Implementation Details

As seen in the previous subsection, we estimate the probability that some box $[0,t_i)$ receives too few or too many points by the sum (taken over the boxes) of the probabilities that the box has too few or too many points. We shall estimate such a failure probability by the sum of the two probabilities that (i) the box receives too few and (ii) too many points. For the case that the box receives too many points, [14, Sect. 3] gives an upper bound on the failure probability of

$$\exp(-c_i^+ W_i^+) \prod_{j=0}^{N-1} \left(\sum_{k=1}^{m} \widetilde{x}_{jk} \exp(1_{[0,t_i)}(y_j, \hat{k}) c_i^+) \right), \tag{8}$$

where c_i^+, W_i^+ and $\widetilde{x}_{jk}$ are as follows.

By $\widetilde{x}_{jk}$ we denote the expected value of the random variable X_{jk} if it has not been rounded, or the outcome of the rounding thereafter. Here and in the following, probabilities, expectations etc. refer to the randomized construction which we aim to derandomize. At no occasion, our algorithm will use randomness itself. Let W_i

denote the expected number of points to lie in the box $[0,t_i)$. By construction, it is equal to number of points $N\lambda_s([0,t_i))$ we aim at having in the box $[0,t_i)$. Let $W_i^+ \geq W_i$ be the maximum number of points we want to tolerate in this box. Define δ_i^+ via $(1+\delta_i^+)W_i = W_i^+$ and $c_i^+ := \log(1+\delta_i^+)$.

With these constants, the sum in equation (8) simplifies as follows. Let $y_j \in \mathscr{P}_{s-1}$ be such that y_j lies in the projection of $[0,t_i)$ to the first $s-1$ coordinates. If $(X_{jk})_{k=1}^m$ is unrounded, then $\sum_{k=1}^m \widetilde{x}_{jk} \exp(1_{[0,t_i)}(y_j,\hat{k})c_i^+) = 1+\delta_i^+ t_i(s)$, where $t_i(s)$ is the last coordinate of t_i. If $(X_{jk})_{k=1}^m$ is rounded, this sum evaluates to either $1+\delta_i^+$ or 1, depending on whether the jth point of $\mathscr{P}_s$ was placed inside box $[0,t_i)$ or not.

The case that the box receives too few points is not covered by Raghavan, but can be treated similarly by regarding the probability that the complement of the box receives too many points. Ignoring points that already miss box $[0,t_i)$ due to an earlier coordinate, the formulas work out the same, with $t_i(s)$ replaced by $1-t_i(s)$, and with a different error tolerance δ_i^-.

Thus we can compute the pessimistic estimator efficiently: when determining the rounded value for $(\widetilde{x}_{jk})_{k=1}^m$ for some j, we only need to replace the terms involving these variables with the m_s possible choices for $(\widetilde{x}_{jk})_{k=1}^m$. This can be done quite efficiently in time $O(nm_s)$, and both computing the initial value of the estimator and computing *all* subsequent values takes time $O(nNm_s)$.

The discussion so far works for all values of δ_i^+ and δ_i^-, provided the initial estimator evaluates to less than one. Potentially, since these represent guarantees on the star discrepancy of $\mathscr{P}_s$, they seem to provide many opportunities for minimizing the resulting discrepancy by setting them according to a desired total discrepancy bound. In practice, however, we observed best results by choosing each δ_i^+ and δ_i^- in such a way that the corresponding failure probability was just less than $1/(2n)$. The values for δ_i^+ and δ_i^- leading to these failure probabilities can be approximated conveniently via binary search.

To compute the discrepancy of the resulting point set, we could track the updates of the pessimistic estimators (since, as noted above, they contain a factor $1+\delta_i^+$ for each point placed inside the corresponding box). However, since the points are placed along a grid of reasonable size, it is just as practical to calculate the discrepancy explicitly in a naive way, that is, looking at all boxes constructible from the coordinates used by the point set, as described in [2, equation (4.1)] or [7, equation (1)].

3 Numerical Experiments

In our numerical experiments we use the star discrepancy of an output set as measure of quality. Calculating the actual star discrepancy of an arbitrary given set is a difficult problem, which was proved to be *NP*-hard in [7]. Also all known algorithms that approximate the star discrepancy up to some user-specified error have a run-time exponential in s and only a very limited range of application, see [17, 18, 6] and the references therein. If we construct an s-dimensional point set component by

component via the CBC algorithm starting in dimension 1, then the output set is a subset of the relatively small grid $\mathscr{G}_s$ and its exact discrepancy can be calculated without essentially increasing the effort. But if we start in some dimension $1 \leq s' < s$ with a given point set and extend it via the CBC algorithm to an s-dimensional point set (cf. Sect. 3.5) or if we randomize the output set (cf. Sect. 3.4), then the exact calculation of the star discrepancies will in general be infeasible. That is why we have to use in such cases different estimators of quality.

As for the memory and time requirements of the algorithm, and more generally the instance sizes for which our algorithm is feasible, the answer depends on whether memory or time is the critical issue. Since the algorithm very frequently needs to ask the question "does point number i lie inside box number j?", it speeds up the execution time significantly to store the answers to these questions in a data structure. If this is done, then the key issue becomes one of memory usage—at $s = 10$ and 500 points, the execution time on our computation server is roughly twenty minutes, but the memory requirements exceed 2 GB. On the other hand, without such caching, the running time for the same instance setting rises to need several hours. Going up from 500 to 1000 points at $s = 10$ scales the number of grid boxes by a factor of fifteen, and so would be infeasible with both our variants. In the rest of this section, when we talk about infeasible instances, we mean that the memory requirements for the caching method significantly exceed 2 GB. Our computation server is equipped with AMD Opteron 2220 SE 2.8 GHz processors and 16 GB of memory.

3.1 Dependence on the Dimension

An obvious disadvantage of the CBC approach of [2] compared to the multidimensional approach of [1] is how the discrepancy guarantee depends on the dimension s. While the latter has a discrepancy guarantee roughly proportional to $s^{1/2}$, the discrepancy guarantee of the CBC approach contains a factor of $s^{3/2}$.

To see to which extent the point sets actually display this disadvantage, we computed point sets of 1000 points each in dimension $s = 2, \ldots, 8$ via the CBC algorithm. Their discrepancies together with the theoretical guarantees proven in [2] are depicted in Figure 1. We immediately see that the actual discrepancies are not of order $s^{3/2}$, but rather depend linearly on the dimension.

3.2 Analysing the Discrepancy: Rounding Error vs. Placement Error

The description of the CBC approach in the previous section reveals that the increase of the discrepancy in a single iteration stems from two causes. One is the rounding error inflicted by the derandomized rounding procedure. This is the term $\frac{1}{N}||A(x) -$

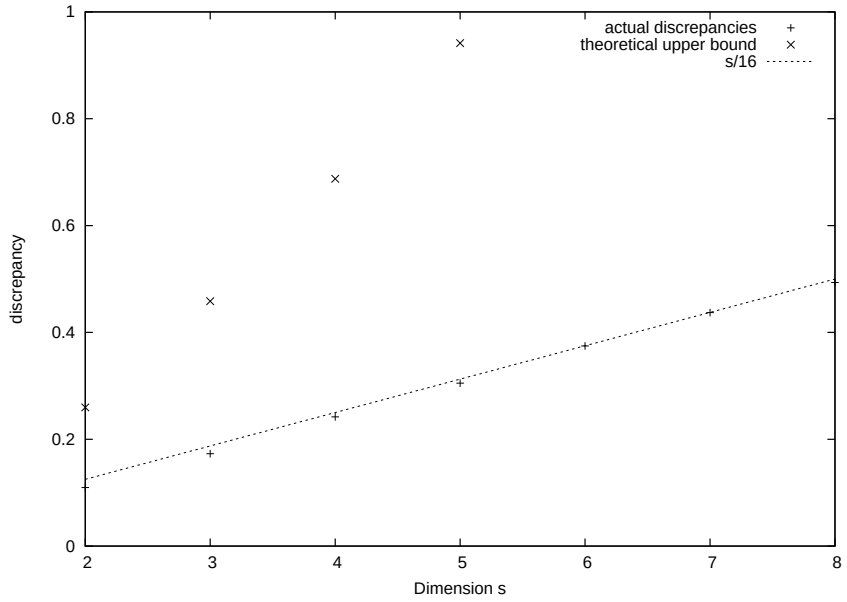

Fig. 1 Predicted and actual discrepancies of 1000 points constructed via the CBC approach.

$A(\bar{x})||_\infty$ in (6), which is of order $O(\sqrt{\log(|\mathscr{G}_s^*|)/N})$, where we recall that $|\mathscr{G}_s^*| = \prod_{d=1}^{s} m_d$. The other is the additional error stemming from placing the points on the grid G_s in the sth coordinate, that is, by putting each point in the middle of an interval defined by G_s. This error was bounded by $1/(2m_s)$ in (6).

If we regard the outcome of all iterations, then the notion of *rounding error* naturally becomes the maximum discrepancy in a box aligned with the grid $\mathscr{G}_s^*$. We define

$$R(\mathscr{P}_s) := \max_{x \in \mathscr{G}_s^*} |\Delta_s(x, \mathscr{P}_s)|. \tag{9}$$

The additional error stemming from placing the points in the center of the grid cells of $\mathscr{G}_s^*$ (*placement error*) can be bounded by what we shall call the *grid gap*

$$\Delta(m_1, \ldots, m_s) := 1 - \prod_{d=1}^{s} (1 - 1/(2m_d)).$$

Indeed, if $\mathscr{P}_s^*$ is an arbitrary N-point set in $[0,1)^s$, and if $\mathscr{P}_s$ is the set we get after translating each point of $\mathscr{P}_s^*$ into the center of the grid cell of $\mathscr{G}_s^*$ it is contained in, then we get

$$D_N^*(\mathscr{P}_s) - \Delta(m_1, \ldots, m_s) \le D_N^*(\mathscr{P}_s^*) \le D_N^*(\mathscr{P}_s) + \Delta(m_1, \ldots, m_s).$$

In [2], the grid widths m_d were chosen in a way that the estimates for the rounding error and the placement error in a single iteration were roughly equal, and hence

their sum was nearly minimized. Note that a finer grid naturally reduces the placement error, but at the same time increases the rounding error due to the $\sqrt{\log |\mathscr{G}_s^*|}$ term in the error bound (which is known to actually occur in many rounding problems). Since the actual discrepancies are much smaller than the predicted ones, it makes sense to analyze the contributions that each cause makes, and possibly adjust the trade-off between rounding error and placement error.

To this purpose, let us first note that the estimate for the placement error cannot be improved, and in consequence, that it always is a lower bound for the resulting discrepancy. Indeed, let $\mathscr{P}$ be any subset of the grid $\mathscr{G}_s$ (as, e.g., an output set of the CBC algorithm). Let $z_\varepsilon := (1-1/(2m_1)+\varepsilon, \dots, 1-1/(2m_s)+\varepsilon)$ for a small $\varepsilon > 0$. Then $|\Delta_s(z_\varepsilon, \mathscr{P})| = 1 - \lambda_s([0, z_\varepsilon)) = 1 - \prod_{d=1}^{s}(1-1/(2m_d)+\varepsilon)$. Hence $\lim_{\varepsilon\to 0} |\Delta_s(z_\varepsilon, \mathscr{P})| = \Delta(m_1, \dots, m_s)$. Thus $\Delta(m_1, \dots, m_s)$ is a lower bound of $D_N^*(\mathscr{P})$.

A comparison of the actual discrepancy of point sets $\mathscr{P}_s$ in different dimensions s with the corresponding grid gap $\Delta(m_1, \dots, m_s)$ and the occurring rounding error can be found in Figure 2; the data points relevant to this discussion are those labeled "standard grid".

The surprising results visible in the figure is that for these data points, the discrepancy $D_N^*(\mathscr{P}_s)$ is in every case very close to the trivial lower bound of $\Delta(m_1, \dots, m_s)$. This is good news in the sense that the rounding procedure contributes almost nothing to the discrepancy. Furthermore, it means that an output set $\mathscr{P}_s$ of the CBC algorithm of size N has more or less the same discrepancy as the full grid $\mathscr{G}_s$, whose cardinality is roughly of order $O(N^{s/2}/\sqrt{s!})$ – note that for the interesting case of s not too small and $s \ll N$ the output set $\mathscr{P}_s$ is a sparse subset of the grid $\mathscr{G}_s$.

3.3 Finetuning the Algorithm

In the light of the previous insight, it makes sense to run the algorithm with finer grids than what was proposed in [2]. This would increase the currently almost nonexisting rounding error and reduce the currently dominant grid gap. Unfortunately, since the run-time of our algorithm is linear in the grid size, it also increases the run-time.

Figure 2 presents some data of this type. Besides showing what constructions are possible in reasonable time, we see the following. Clearly, making the grid finer does reduce the grid gap significantly. When taking twice as many grid subdivisions in each dimension, however, the rounding error remains insignificant compared to the grid gap. Of course, the other side of the coin is that the rounding error only increased by that little that we fully take advantage of the finer grid.

For the grid with four times the number of subdivisions, we do observe a visible rounding error, even if it is still small compared to the grid gap. This means that here we are getting closer to the optimal balance of rounding and placement error. Unfortunately, taking an even finer grid was not feasable. Already conducting the

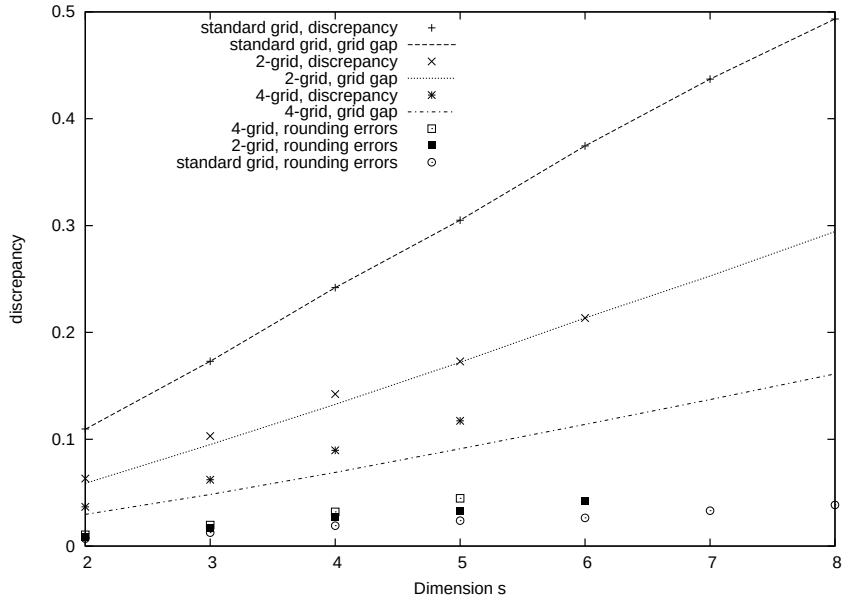

Fig. 2 Discrepancies, grid gaps and rounding errors observed for $N = 1000$ when using the grid proposed in [2] ("standard grid", m_s grid lines in dimension s), and grids with two ("2–grid") and four ("4–grid") times the number of grid subdivisions in each dimension.

$s = 5$ experiment in the last grid took 4.5 hours of computation time (using the caching method, requiring 3 GB of memory; see beginning of Section 3).

3.4 Randomization of the Output Set

In the previous subsection, we saw that taking a finer grid does reduce the grid gap, but at the cost of computation times that quickly make this approach infeasible. Hence taking finer grids generally did not suffice to reduce the grid gap to an extent that the discrepancy guarantee was not dominated by the grid gap.

To overcome this issue, we now propose an additional idea to reduce the placement error and prove superior bounds on it. The idea is simple: Instead of placing the points on the centers of the grid cells, we place them on randomly chosen locations in the grid cell they belong to. Let us make this more precise: We consider the partition $\mathscr{B}^*$ of $[0,1)^s$ into n axis-parallel half-open boxes of equal size with centers in $\mathscr{G}_s$ and right upper corners in $\mathscr{G}_s^*$. We transform the output set $\mathscr{P}_s$ of the CBC algorithm into a set $\mathscr{P}_s^*$ by substituting in each box $B \in \mathscr{B}^*$ the points $p \in \mathscr{P}_s \cap B$ by random points p^* that are independently and uniformly distributed within B. This randomization may enhance the quality of the output set and has the advantage that $\Delta(m_1,\ldots,m_s)$ is not necessarily any longer a lower bound for $D_N^*(\mathscr{P}_s^*)$.

The practical problem that occurs now is that the actual star discrepancy of $\mathscr{P}_s^*$ would be much harder to calculate than the one of the subset $\mathscr{P}_s$ of $\mathscr{G}_s$. Therefore we use an estimator for the discrepancy of $\mathscr{P}_s^*$, which we describe here shortly. To this purpose let us restate a definition and a lemma from [3].

Definition 1. A finite set $\Gamma \subset [0,1]^s$ is called a δ-cover of $[0,1]^s$ if for every $y = (y_1,\dots,y_s) \in [0,1]^s$ there are $x = (x_1,\dots,x_s), z = (z_1,\dots,z_s) \in \Gamma \cup \{0\}$ with $\lambda_s([0,z)) - \lambda_s([0,x)) \le \delta$ and $x_i \le y_i \le z_i$ for all $1 \le i \le s$.

Lemma 1. *Let Γ be a δ-cover of $[0,1]^s$. Then for any N-point set $\mathscr{P}_s \subset [0,1)^s$ we have $D_N^*(\mathscr{P}_s) \le D_N^\Gamma(\mathscr{P}_s) + \delta$, where $D_N^\Gamma(\mathscr{P}_s) = \max_{x\in\Gamma} |\Delta_s(x,\mathscr{P}_s)|$.*

Let $R(\mathscr{P}_s)$ be the rounding error defined in (9). It is not hard to see that $\mathscr{G}_s^*$ is a δ-cover for

$$\delta = 1 - \prod_{d=1}^{s} \left(1 - \frac{1}{m_d}\right).$$

Due to Lemma 1 we get thus $D_N^*(\mathscr{P}_s^*) \le R(\mathscr{P}_s^*) + \delta = R(\mathscr{P}_s) + \delta$. But we can give a much better estimate: For $0 < \delta' \le \delta$ and $p \in (0,1)$ define now

$$\theta = \theta(p,\delta,\delta') = \left(\frac{\delta + 2R(\mathscr{P}_s)}{2N}\right)^{1/2} \left(\log\left(\frac{2}{p}\right) + Q(\delta')\right)^{1/2},$$

where

$$Q(\delta') = \min\left\{ s\log(k), \log\left(\frac{2^s s^s}{s!}\right) + s\log((\delta')^{-1}+1)\right\}$$

and

$$k = k(\delta',s) = \left\lceil \frac{s}{s-1} \frac{\log(1-(1-\delta')^{1/s}) - \log(\delta')}{\log(1-\delta')} \right\rceil + 1.$$

Theorem 1. *$D_N^*(\mathscr{P}_s^*) \le R(\mathscr{P}_s) + \theta(p,\delta,\delta') + \delta'$ with probability at least $1-p$.*

The proof employs Hoeffding's large deviation bound for all test boxes $[0,x)$, with x from a minimal δ'-cover G'. To perform a union bound, the results [3, Thm.2.3] and [6, Thm.1.15] are used to upper-bound the cardinality of G'.

In our tests we chose $p = 0.05$ and used the estimator

$$R(\mathscr{P}_s) + \min_{0<\delta'\le\delta} \left(\theta(0.05,\delta,\delta') + \delta'\right),$$

which is an upper bound for $D_N^*(\mathscr{P}_s^*)$ with probability at least 95%.

Figure 3 shows the resulting estimates for the discrepancies of randomized output sets $\mathscr{P}_s^*$ in dimension $s = 10$ for values of N between 50 and 500, together with the corresponding rounding and placement errors of the related (deterministic) output sets $\mathscr{P}_s$. These experiments strongly indicate that the final (local) randomization procedure enhances the quality of the output sets significantly, since the estimated discrepancies are all clearly smaller than the corresponding grid gaps, which are lower bounds for the discrepancies of the (non-randomized) output sets. Therefore

the randomized CBC method seems to be a promising alternative to the conventional CBC method which overcomes the lower discrepancy bound in form of the grid gap.

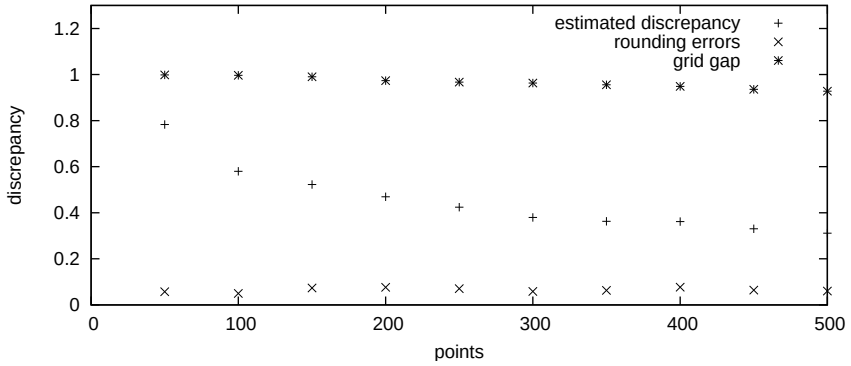

Fig. 3 Comparison of the estimated discrepancy of 10-dimensional randomized output sets with the corresponding rounding errors and grid gaps.

In Table 1 we listed the run-times of the randomized CBC algorithm for some representative values of N. Note that the CBC algorithm used is the version with higher memory requirements (see beginning of Section 3), and that the time needed for the final randomization of the point set is negligible.

Table 1 Times for creating point sets using the randomized CBC method.

N	Running time, $s = 5$	Running time, $s = 10$
100	0.037s	0.598s
300	0.644s	1m45.967s
500	2.196s	20m49.5s

3.5 A Hybrid Approach: Extending Halton-Hammersley Point Sets

One of the strengths of the CBC method is that we may also use it to extend an existing point set to one in higher dimension. This could be a promising idea, because the classical constructions lead to very good point sets in small dimensions. This advantage is used by a related idea called padding quasi-Monte Carlo (QMC) by Monte Carlo (MC) – there a low-discrepancy sequence is extended in the dimension by choosing the additional coordinates randomly, see, e.g., [15, 13, 5]. The resulting point sets show in many applications a better performance than pure QMC or MC points. Here we want to study whether the same is true for extending low-discrepancy point sets by the CBC algorithm. Therefore, we pursue the following.

Fix N, the number of points to be constructed, and s, the dimension the final point set shall have. For some $s' \leq s$, let $\mathscr{P}^*_{s'}$ be a Halton-Hammersley point set of N points in dimension s' (see, e.g., [8] or [12]). Then we perform $s-s'$ CBC iterations to obtain an N-point set $\mathscr{P}_s$ in $[0,1)^s$. We want to stress that in our experiments the decisions of the CBC algorithm which additional components to choose to extend $\mathscr{P}_{s'}$ are only based on the discrepancies induced by all boxes of the form $[0,t)$, $t \in \mathscr{G}^*_d$, $d = s'+1,\ldots,s$. (In general it would be possible to substitute $\mathscr{G}^*_d$ by a finer grid whose projection onto the first s' components is the grid generated by the coordinate values of the points of $\mathscr{P}_{s'}$. But such an s'-dimensional grid is already of size $\Theta(N^{s'})$ and will therefore increase the computation time crucially.) If $s' \geq 1$, the resulting point set $\mathscr{P}_s$ is not any longer a subset of the (relatively small) grid $\mathscr{G}_s$. For $s = 10$ and N in the hundreds this means practically that calculating the exact discrepancy of $\mathscr{P}_s$ is in general infeasible. Therefore we restrict ourselves to computing the rounding errors of the resulting point sets, as defined in (9).

We performed our experiments for $s = 10$. The results for $s' = 0$ (the pure CBC approach), 3, 5, 7 and 10 (the pure 10–dimensional Halton-Hammersley set) and $N = 50, 100, 150, \ldots, 500$ are depicted in Figure 4.

The results support our theoretical considerations. For small numbers of points, the relatively large discrepancy of high-dimensional Halton-Hammersley sets leads to inferior results for large values of s'. For larger numbers of points, the pure Halton-Hammersley set is not very good, but becomes better than the pure CBC construction. Better results are obtained for the intermediate values $s' = 3, 5, 7$.

4 Acknowledgments

The authors thank the editor Art B. Owen and two anonymous referees for their comments, which greatly helped to improve the presentation. Benjamin Doerr and Magnus Wahlström gratefully acknowledge support from the German Science Foundation (DFG) via its priority programme "SPP 1307: Algorithm Engineering", grant DO 479/4-1. Michael Gnewuch gratefully acknowledges support from the DFG under grant GN 91/3-1.

References

1. Doerr, B., Gnewuch, M.: Construction of low-discrepancy point sets of small size by bracketing covers and dependent randomized rounding. In: A. Keller, S. Heinrich, H. Niederreiter (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2006, pp. 299–312. Springer-Verlag, Berlin Heidelberg (2008)
2. Doerr, B., Gnewuch, M., Kritzer, P., Pillichshammer, F.: Component-by-component construction of low-discrepancy point sets of small size. Monte Carlo Methods Appl. **14**, 129–149 (2008)
3. Doerr, B., Gnewuch, M., Srivastav, A.: Bounds and constructions for the star discrepancy via δ-covers. J. Complexity **21**, 691–709 (2005)

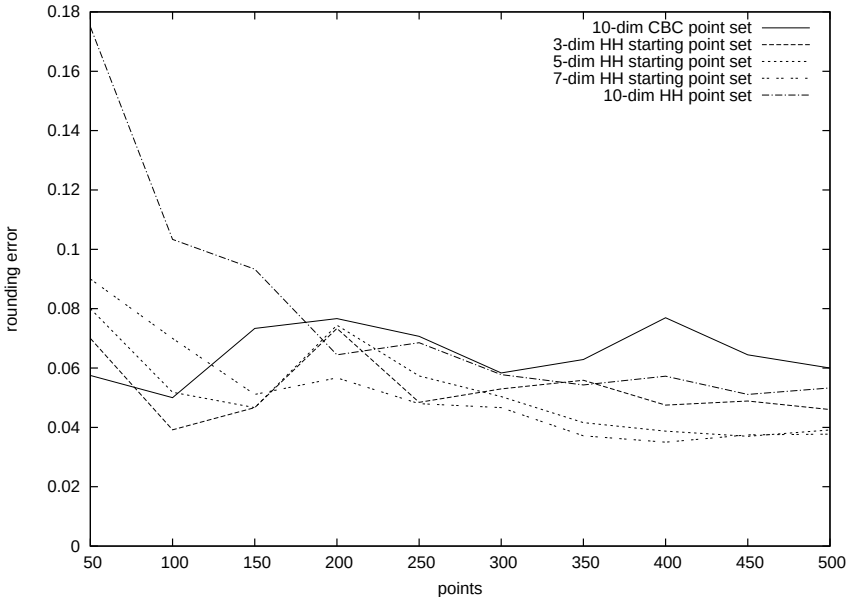

Fig. 4 Rounding errors of 10-dimensional point sets stemming from different versions of the hybrid approach (starting with a 3, 5 and 7 dimensional Halton-Hammersley set and then adding dimensions via the CBC approach), together with the two pure approaches.

4. Doerr, B., Wahlström, M.: Randomized rounding in the presence of a cardinality constraint. In: Proceedings of ALENEX, pp. 162–174. SIAM (2009)
5. Gnewuch, M.: On probabilistic results for the discrepancy of a hybrid-Monte Carlo sequence. J. Complexity, 2009. (http://dx.doi.org/10.1016/j.jco.2009.02.009)
6. Gnewuch, M.: Bracketing numbers for axis-parallel boxes and applications to geometric discrepancy. J. Complexity **24**, 154–172 (2008)
7. Gnewuch, M., Srivastav, A., Winzen, C.: Finding optimal volume subintervals with k points and calculating the star discrepancy are NP-hard problems. J. Complexity **25**, 115–127 (2009)
8. Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multidimensional integrals. Numer. Math. **2**, 84–90 (1960)
9. Heinrich, S.: Some open problems concerning the star-discrepancy. J. Complexity **19**, 416–419 (2003)
10. Heinrich, S., Novak, E., Wasilkowski, G.W., Woźniakowski, H.: The inverse of the star-discrepancy depends linearly on the dimension. Acta Arith. **96**, 279–302 (2001)
11. Hickernell, F.J., Sloan, I.H., Wasilkowski, G.W.: On tractability of weighted integration over bounded and unbounded regions in $\mathbb{R}^s$. Math. Comp. **73**, 1885–1901 (2004)
12. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods, *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*, vol. 63. SIAM, Philadelphia (1992)
13. Ökten, G., Tuffin, B., Burago, V.: A central limit theorem and improved error bounds for a hybrid-Monte Carlo sequence with applications in computational finance. J. Complexity **22**, 435–458 (2006)
14. Raghavan, P.: Probabilistic construction of deterministic algorithms: Approximating packing integer programs. J. Comput. Syst. Sci. **37**, 130–143 (1988)
15. Spanier, J.: Quasi-Monte Carlo methods for particle transport problems. In: H. Niederreiter, P.J.S. Shiue (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, pp. 121–148. Springer-Verlag, Berlin (1995)

16. Spencer, J.: Ten Lectures on the Probabilistic Method. SIAM, Philadelphia (1987)
17. Thiémard, E.: An algorithm to compute bounds for the star discrepancy. J. Complexity **17**, 850–880 (2001)
18. Thiémard, E.: Optimal volume subintervals with k points and star discrepancy via integer programming. Math. Meth. Oper. Res. **54**, 21–45 (2001). Extended version available at http://ina2.eivd.ch/Collaborateurs/etr/