Spoke Darts for Efficient High Dimensional Blue Noise Sampling

Mohamed S. Ebeida, Sandia National Laboratories
Scott A. Mitchell, Sandia National Laboratories
Muhammad A. Awad, Alexandria University
Chonhyon Park, UNC Chapel Hill
Laura P. Swiler, Sandia National Laboratories
Dinesh Manocha, UNC Chapel Hill
Li-Yi Wei, Univ. Hong Kong

Abstract

Blue noise refers to sample distributions that are random and well-spaced, with a variety of applications in graphics, geometry, and optimization. However, prior blue noise sampling algorithms typically suffer from the curse-of-dimensionality, especially when striving to cover a domain maximally. This hampers their applicability for high dimensional domains.

We present a blue noise sampling method that can achieve high quality and performance across different dimensions. Our key idea is spoke-dart sampling, sampling locally from hyper-annuli centered at prior point samples, using lines, planes, or, more generally, hyperplanes. Spoke-dart sampling is more efficient at high dimensions than the state-of-the-art alternatives: global sampling and advancing front point sampling. Spoke-dart sampling achieves good quality as measured by differential domain spectrum and spatial coverage. In particular, it probabilistically guarantees that each coverage gap is small, whereas global sampling can only guarantee that the sum of gaps is not large. We demonstrate advantages of our method through empirical analysis and applications across dimensions 8 to 23 in Delaunay graphs, global optimization, and motion planning.

Keywords: blue noise, sampling, high dimension, Delaunay graph, global optimization, motion planning

1 Introduction

Blue noise refers to sample distributions that are random and well-spaced. These are desirable properties for sampling in computer graphics, as randomness avoids aliasing while uniformity reduces noise. Blue noise sampling has been applied to a variety of graphics applications, such as rendering [Cook 1986; Sun et al. 2013; Chen et al. 2013], imaging [Balzer et al. 2009; Fattal 2011; de Goes et al. 2012], geometry [Alliez et al. 2003; Öztireli et al. 2010], animation [Schechter and Bridson 2012], visualization [Li et al. 2010], and numerical computation [Ebeida et al. 2014b].

Both the random and well-spaced properties are agnostic with respect to the dimensionality of the underlying sample domain. Thus, both high and low dimensional applications can and should benefit from blue noise. However, most existing applications are limited to low dimensions, predominantly 2-d, in part because of the difficulty of producing good blue noise distributions in high dimensions. For example, blue noise has been deployed to construct meshes for 2-d and 3-d domains, with questionable practicality for much higher dimensions; in global optimization, point placement in higher dimensions becomes a critical issue to efficiently explore the parameter space; in robotic planning, blue noise has been used for configuration spaces up to 6-d [Park et al. 2013] but not in higher dimensions for more complex or realistic agents and environments.

A key reason mostl results are low dimensional is the curse of dimensionality — prior blue noise algorithms do not scale well to high dimensions, especially when striving to fill in the domain maximally [Ebeida et al. 2011]. No prior algorithm can guarantee local saturation within tractable runtime. The algorithms closest to obtaining this goal are based on advancing-front [Liu 1991; Bridson 2007] or k-d darts [Ebeida et al. 2014b]. Our method combines these two approaches.

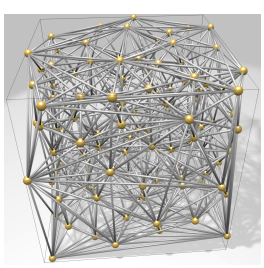
Advancing-front methods generate samples locally from the distribution boundary and gradually advanced towards the rest of the domain. Building the geometric boundaries explicitly [Liu 1991] through the intersections of sample disks results in a combinatorial explosion in complexity in high dimensions. Advancing-Front Point dart-throwing (AFP) [Bridson 2007] avoids building the front geometry. Each accepted sample has a disk around it that rejects future samples. For each sample, AFP does rejection sampling from an annulus around its disk, and proceeds to the next sample after a fixed number (30) of consecutive rejections. Its advantage is locality. The advancing front mitigates the effects of domain size. It ensures that the saturation properties around the current sample also apply to future and past samples. Its disadvantage is that point-rejection yields a volume-fraction saturation guarantee which decreases exponentially with dimension; worse, this guarantee applies only within the annulus and *there is no known bound on the void volume outside all annuli.*

k-d darts [Ebeida et al. 2014b] selects samples using hyperplanes: select a random axis-aligned hyperplane, find its uncovered subset, and select a point from this subset. A rejection occurs only when the entire hyperplane is covered. Its advantage is that hyperplanes mitigate the effects of high dimensions, because rejection is much less likely than for point samples. Its disadvantage is that it does not provide local saturation, because hyperplanes are selected globally from the entire domain.

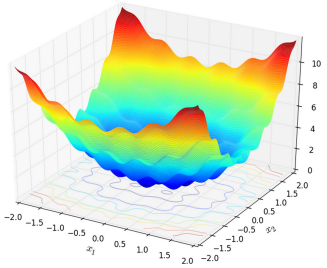
Our method achieves guaranteed local saturation within tractable runtime. Our key idea is to combine the advantages of the two prior methods: the local saturation of AFP and the dimension-mitigation of k-d darts. Specifically, our method replaces the point-sampling of AFP with hyperplane sampling, especially line sampling.

We call our method *spoke-dart sampling*. A *spoke-dart* is a set of *spokes* passing through a prior sample point. Each spoke is a hyper-annulus, such as 1-d line segments or a 2-d planar ring as illustrated in Figure 2. In contrast to constructing the front explicitly, we trim each spoke by existing sample disks, and select the next sample from the remaining regions. Since each spoke has a local scope, the trimming can be efficiently performed locally instead of globally as in k-d darts.

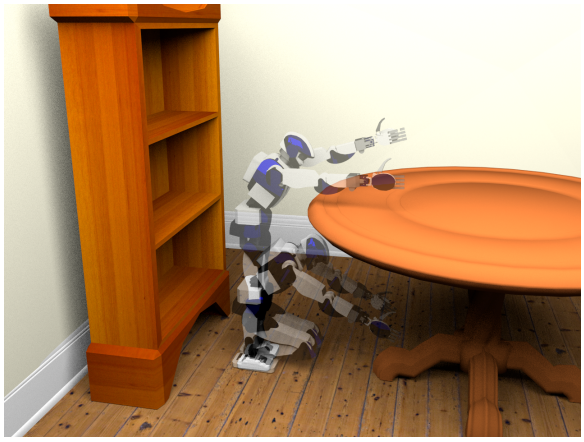
The advancing front nature also helps our method saturate the domain better than prior dimensionality-agnostic methods such as brute-force dart throwing [Cook 1986] or AFP. Moreover, global methods such as dart throwing and k-d darts only provide global saturation. If parameters are chosen so that a global method and spoke-dart sampling produce the same total gap volume, spoke-dart sampling will likely have that volume distributed in small gaps throughout the domain, whereas a global method might have the gap volume concentrated in one large component. Thus spoke-dart sampling ensures that the maximum distance from a domain point to its nearest sample, r_c, is smaller. With high probability, $(1-\epsilon)$, it achieves the user-desired saturation, as measured by the desired ratio $\beta = r_c/r_f$ between the coverage radius r_c and conflict radius



(a) *Delaunay graph*



(b) *global optimization*



(c) *motion planning*

Figure 1: *Spoke-dart sampling is an effective method for high-dimensional sampling with applications in Delaunay graph construction (a), global optimization (b), and motion planning (c).*

r_f of sample disks [Mitchell et al. 2012a].

Spoke-dart sampling has polynomial (instead of exponential) time complexity with respect to the sample-space dimension d and number of samples n. The only exponential-in-d dependency is when high saturation is desired. The time is $O\left(d(-\log\epsilon)(\beta-1)^{1-d}n^2\right)$; if $\beta = 2$ and a fixed epsilon (say 10^{-5}) is desired, then this reduces to $O(dn^2)$. We do not know of explicit formulas for the run-time needed for dart-throwing and k-d darts to achieve a given local β, but for large domains it appears that they scale much worse than spoke-dart sampling. Another key advantage of spoke-dart sampling is it requires only *linear* memory, $O(dn)$, regardless of saturation β. This is in sharp contrast to grid-based methods such as Simple MPS [Ebeida et al. 2012], where the memory is exponential, $O(2^d)$, when the top level grid is first refined; and careful management is needed to avoid the $O(((-\log_2(\beta-1))^d)$ memory for naive grid refinement. To our knowledge, we provide the first feasible method to produce probabilistically-guaranteed locally-saturated blue-noise point-sets in high dimensions.

We apply spoke-dart sampling to high dimensional Delaunay graphs, global optimization, and motion planning; see Figure 1. We generate an approximate Delaunay graph in high dimensions, where the exact version is too expensive to generate. For global optimization, the well-known DIRECT algorithm [Jones et al. 1993] for Lipschitzian optimization might not scale well to high dimensions due to its hyperrectangular sample neighborhoods and deterministic patterns. Our method places hyperspheres stochastically, and significantly improves convergence speed. For motion planning, a common method is RRT (Rapidly-exploring Random Tree) [Kuffner and Lavalle 2000]. MPS has been used in RRT for up to six dimension [Park et al. 2013]. Our method can efficiently handle configuration spaces with more than 20 dimensions.

The contribution of this paper can be summarized as follows:

- The idea of *spoke-dart sampling,* which combines the advantages of state-of-the-art methods: the locality of advancing-front and the dimension-mitigation of k-d darts;
- Efficient trimming and sampling algorithms for spoke-darts in different dimensions;
- Empirical and theoretical bounds for key measures such as time and memory complexity, and β for coverage (a.k.a. saturation, maximality, or well-spacedness);
- Applications in high dimensional Delaunay graphs, global optimization, and motion planning.

2 Background

Blue noise sampling algorithms and applications have much prior art. Here we focus on those most relevant to high dimensionality, maximal sampling, non-point samples, and advancing front.

High dimensionality The curse-of-dimensionality refers to the difficulty of obtaining efficient data-structures and algorithms in high dimensions. The goal is to avoid time and memory complexities that have higher than polynomial growth.

Examining nearby samples (e.g. for conflict) is a key step among all known blue noise methods. However, the number of potential neighbors grows exponentially with dimension d, related to the mathematical "kissing number." Finding them efficiently is also difficult. Neighborhood queries is an active research topic in computational geometry [Samet 2005; Arya and Mount 2005; Miller et al. 2013]. In high dimensions, it is hard to be more efficient than examining every point. The good news is that this is only linear in the output size.

Maximal sampling A maximal disk sampling is fully saturated and can receive no more samples. As a consequence the distribution is well-spaced. Such saturation is important for many applications, as described in the extensive literature on Maximal Poisson-disk Sampling (MPS) [Cline et al. 2009; Gamito and Maddock 2009; Ebeida et al. 2011; Ebeida et al. 2012; Yan and Wonka 2013].

There are practical algorithms that achieve maximality in low dimensions, but none do so in high-dimensions. The maximal methods for low dimensions use data structures (e.g. grids or trees) which do not scale well beyond six dimensions, as surveyed in Ebeida et al. [2014b]. Brute force dart throwing [Dippé and Wold 1985; Cook 1986] scales easily to high dimensions, but does not reach maximality even in low dimensions. The notion of a "relaxed MPS" [Ebeida et al. 2014b], one that is measurably saturated but short of maximal, is attractive for higher dimensions. Our method achieves a form of relaxed MPS and is more efficient than the prior k-d darts method [Ebeida et al. 2014b].

Two-radii MPS methods [Mitchell et al. 2012b; Ebeida et al. 2014a] provide a way to adjust and measure the saturation of Poisson-disk distributions. The key idea is to define two radii around each sample, r_c for domain coverage and r_f for inter-sample distances. In particular, r_c is the maximum distance between a domain point and its nearest sample, and r_f is the minimum distance between two samples. Their ratio $\beta = r_c/r_f$ quantifies saturation, and affects

the randomness and well-spacedness of the distribution. We use β as a control parameter, as do some prior works. Ebeida et al. [2013] is a post-processing algorithm that can reduce the number of sample points while preserving $\beta = 1$.

Beyond point samples k-d darts [Ebeida et al. 2014b] is the state-of-the-art for high-dimensional relaxed Poisson-disk sampling, and uses axis-aligned hyperplanes to find regions of interest. A hyperplane is especially efficient for dealing with arrangements of spheres, because its intersection with a sphere is a simple analytic sphere in the lower dimensional space of the hyperplane. However, k-d dart hyperplanes extend throughout the entire domain, and it can be expensive to intersect them with densely sampled subregions. Here we seek to avoid global computation. Our method is essentially a local, advancing front version of the relaxed MPS method in k-d darts [Ebeida et al. 2014b].

Sun et al. [2013] samples lines and line-segments for rendering applications, including 3-d motion blur, 4-d lens blur, and 5-d temporal light fields. For determining sample positions it relies on subroutines that do not scale well to high dimensions.

Advancing front Advancing front methods were initially proposed for meshing [Liu 1991; Li et al. 1999; Liu et al. 2008] and later adopted for sampling in graphics [Dunbar and Humphreys 2006]. The basic idea is to draw new samples from boundaries (fronts) of existing sample sets and gradually expand towards the rest of the domain. These methods are designed mainly for low dimensions and can run fast in 2-d and 3-d. However, they maintain geometric and combinatorial structures for the fronts which do not scale well to high dimensions. Our method is based on advancing front, but maintains and samples from an implicit front, to avoid this combinatorial complexity.

3 Method

Our spoke-dart sampling method builds and improves upon both advancing front and k-d darts. Similar to advancing front, our method generates new samples from the current sample set boundary and gradually expands towards the rest of the domain. The key differences between our method and prior methods are (1) how the front is constructed and (2) how new samples are drawn. In particular, prior methods compute fronts and samples from intersections of existing sample disks which can be quite complex, whereas our method uses spoke-darts with very simple structures.

Similar to k-d darts [Ebeida et al. 2014b], spoke-darts are sets of k-dimensional hyperplanes. However, unlike k-d dart hyperplanes which are global and axis-aligned, spokes are local hyper-annuli and randomly oriented. A spoke-dart passes through an extant sample. Such locality and randomness properties given spoke-darts computational advantages over k-d darts.

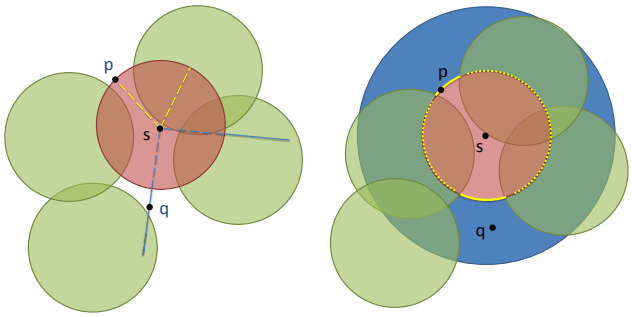
Below, we first describe the basic representation and operations for spoke-darts, followed by how we use them for blue noise sampling.

3.1 Representation

A spoke-dart is a set of randomly-oriented hyper-annuli that pass through a given sample point s. Each such annulus is a spoke with radius spanning $[r, r + w]$, where w is its size (width) and r is its starting distance away from the sample. For blue noise we use r equal to the Poisson disk radius, and for some later applications we use $r = 0$. Spokes can have various dimensions, up to the dimension of the sample domain. For example, a 2-d domain can have 1-d line-spokes and 2-d plane-spokes, as illustrated in Figure 2. Line-spokes are line segments starting from a random point on the disk $D(s)$ of a given sample s and extending in the radial direction for a distance of w. Similarly, a plane-spoke is an annulus starting from a random great circle on $D(s)$ and extending in the radial direction for a distance of w.

Spokes can be degenerate in the sense that they have $w = 0$. A *degenerate line-spoke* reduces to a point, and a *degenerate plane-spoke* reduces to a great circle, on $D(s)$.

Spokes with three or higher dimensions can be defined analogously. In principle spoke-dart sampling works with spokes of any dimension. Since we have used only line-spokes and plane-spokes in our current implementation, we describe only those below.



(a) *line-spokes* **(b)** *plane-spokes*

Figure 2: *Spokes in a 2-d domain. In (a), a line-spoke (blue) is a radial line segment of length w starting from the disk surface $D(s)$ of a sample s. Degenerate line-spokes (yellow) have zero length and lie on the disk surface. In (b), a plane spoke (blue) is a planar annulus. Degenerate plane-spokes (yellow) are circles on the disk. Here p and q indicate example samples drawn from degenerate and full spokes.*

3.2 Operations

As in Poisson-disk sampling, spoke-darts are used to explore the gaps or *voids*, the uncovered space that can accept a new sample. Specifically, we *trim* a spoke with existing sample disks, and select a new sample from the remaining uncovered hyperplane region(s). The key to our design is that all trimming operations are efficient in high dimensions, even with many nearby sample disks. In particular, we leverage the degenerate versions of spokes to avoid wasting time while trimming their full versions.

Spoke generation A random line-spoke is generated by selecting its starting position p on the disk surface $D(s)$ uniformly by area. A random plane-spoke is generated by selecting a great circle with uniform random orientation, by selecting two such points that the plane passes through. To generate p, we generate each of its d coordinate independently from a normal (Gaussian) distribution. Then we linearly scale the vector of coordinates to the disk radius [Muller 1959]. (For $r = 0$ spokes, p defines direction only.) In our current implementation, w is initialized by a global constant.

Line-spoke trimming Our line-spoke trimming method is summarized in Algorithm 1. Our implementation uses two simplifications for efficiency, to avoid the problem that trimming a full spoke directly by nearby disks can involve a lot of wasted effort. There could be many (kissing number) nearby disks.If we iterate over the disks, many segments are generated early on, which are completely

covered later. Often there is nothing left, but it took many operations to discover this. Our first simplification is to check if the degenerate version of a line-spoke (a point) is covered before trimming its full version. This involves no partitioning, and it is discarded by the first disk we find that covers it. It is possible that we might miss an uncovered segment of a line-spoke, but overall it is more efficient. The second simplification is to keep only the line segment that touches $D(s)$, and ignore the rest of the partition. Hence each trimming operation shortens the length of the line-spoke instead of fragmenting it into pieces. This favors inserting nearer samples, but the net effect is small; see Figure 4.

Input: input line spoke ℓ_1 at sample s
Output: trimmed spoke ℓ_1'
$\ell_1' \leftarrow \ell_1$
for each sample s' near s **do**
  $\ell_1' \leftarrow \ell_1' - D(s')$
  $\ell_1' \leftarrow$ only the segment of ℓ_1' touching $D(s)$
  if ℓ_1' is empty **then**
    return empty
return ℓ_1'

Algorithm 1: *Trimming a line spoke.*

Input: input plane spoke ℓ_2 at sample s
Output: trimmed line spoke $\ell_1 \subset \ell_2$
$\ell_2^0 \leftarrow$ `Degenerate`(ℓ_2) `// circle`
$p \leftarrow$ `RandomSample`(ℓ_2^0)
while p has not traversed a full revolution of ℓ_2^0 **do**
  if p is covered by a disk $D(s')$ **then**
    $p \leftarrow$ `LeftIntersection`$(\ell_2^0, D(s'))$
  else
    $w \leftarrow$ thickness of ℓ_2
    $\ell_1 \leftarrow$ `LineSpoke`(p, w)
    return `TrimLineSpoke`(ℓ_1) `// Algorithm 1`
return empty `//` ℓ_2^0 `was covered`

Algorithm 2: *Trimming a plane spoke.*

Plane spoke trimming We apply similar concepts for trimming plane-spokes as summarized in Algorithm 2. For efficiency we start with a degenerate spoke, a circle. We search for an uncovered point on its circle as follows. Take any point p from the circle; a point we used to generate the plane is a good choice. If p is covered by a disk, we move p "left" along the circle to the intersection point of the circle and that disk. We repeat this until p is uncovered; or we have passed our starting point, in which case the circle is completely covered. For a full plane-spoke, we now throw a line-spoke passing through p and trim it as in Algorithm 1.

3.3 Sampling

With the representation and operations of spoke-darts above, our blue noise generation method works as follows. We initialize the output set with one random sample and put it into the active pool of front points. When this pool becomes empty our algorithm terminates. We remove a random sample s from the pool and try to generate new samples s' from random spokes ℓ through s. Accepted samples are added to the pool. We keep throwing spokes from the same sample s until m consecutive spokes failed to generate an acceptable sample. Our method is summarized in Algorithm 3.

Input: sample domain Ω
Output: output sample set $\mathcal{S}$
$s \leftarrow$ `RandomSample`(Ω)
$\mathcal{S} \leftarrow \{s\}$
$\mathcal{P} \leftarrow \{s\}$ `// active pool`
while $\mathcal{P}$ not empty **do**
  $s \leftarrow$ `RandomSelectAndRemove`$(\mathcal{P})$
  $reject \leftarrow 0$
  while $reject \leq m$ **do**
    $\ell \leftarrow$ `RandomSpoke`$(s, r, r + w)$
    $\ell \leftarrow$ `Trim`(ℓ) `// Algorithm 1 or 2`
    if ℓ not empty **then**
      $s' \leftarrow$ `RandomSample`(ℓ)
      $\mathcal{S} \leftarrow \mathcal{S} \bigcup \{s'\}$
      $\mathcal{P} \leftarrow \mathcal{P} \bigcup \{s'\}$
      $reject \leftarrow 0$
    else
      $reject \leftarrow reject + 1$
return $\mathcal{S}$

Algorithm 3: *Sampling with spoke-darts.*

3.4 Implementation

Note that for each sample on the front, for each spoke-dart we iterate over the nearby samples, at distances less than $3r$. It saves time to collect a sample's neighbors once before throwing any of its spokes, if the number of neighbors happens to be less than the total number of points, $N < n$. Those wishing to reproduce our output may simply iterate over all the points and gather these neighbors in an array. In our implementation, we have found that using a k-d tree saves time in moderate dimensions.

We maintain a k-d tree of the entire point set. We collect the subtree of neighbors. We update them as we successfully add new disks. If the neighbor list is huge, $N \rightarrow n$, as can happen when d is very large, then these trees do not save any time over an array, but they are not significantly more costly either. (None of our run-time proofs depend on these trees.)

4 Analysis

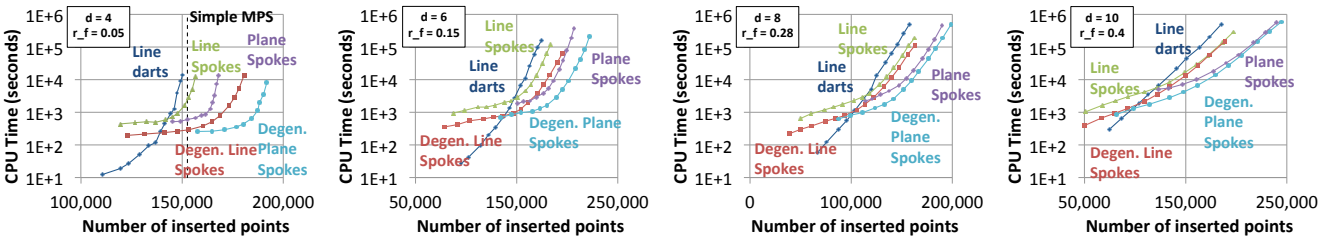
Here we compare our method against the state-of-the-art, and analyze the quality and performance of the variations of our method: $\{\text{line, plane}\} \times \{\text{full, degenerate}\}$ spokes.

To the best of our knowledge, k-d dart [Ebeida et al. 2014b] is the state-of-the-art method for high dimensional blue noise sampling with high saturation, so we compare to it. For dimensions below six, we may compare to MPS output produced by the Simple MPS algorithm [Ebeida et al. 2012].

4.1 Performance Analysis

To consider both speed and saturation, we measure the accumulated computation time with respect to the number of generated samples across different dimensions for all candidate methods. Since the distributions generated by the different methods are not the same, especially for degenerate and non-degenerate spokes, equal number of points does not mean equal saturation.

k-d dart versus spoke-darts As shown in Figure 3, when the domain is relatively empty, line darts [Ebeida et al. 2014b] are better than our methods. However, when the domain is relatively full,



(a) $d = 4$ **(b)** $d = 6$ **(c)** $d = 8$ **(d)** $d = 10$

Figure 3: *Comparison between k-d dart and different variations of our method under different dimensions. Each result is sampled from a domain is a d-dimensional unit box domain with Poisson disk spacing r_f. The goal is to fill in the domains with as many samples as possible under the same amount of computation time. Note that towards the end game with higher fill-rates, our methods consistently outperform k-d dart, with plane-spokes outperforming line-spokes, and the degenerate versions outperforming the full versions.*

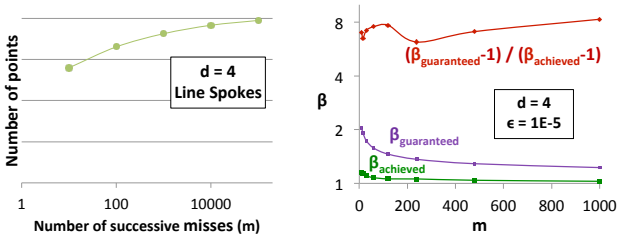
our methods are better. It appears that after a critical level of saturation, line darts nearly stall, while spoke-darts continue to add points. These thresholds depend on the dimension and radius. Figure 3 shows that if one desires a highly saturated (low β) distribution, our method is orders of magnitude faster. These empirical results are consistent with our earlier theoretical observations about the global versus local nature of k-d darts and spoke-dart sampling.

Variations among spoke-darts Figure 3 also points towards the general trends that degenerate spokes are more efficient than full spokes, and plane-spokes are more efficient than line-spokes, at producing points.

4.2 Quality Analysis

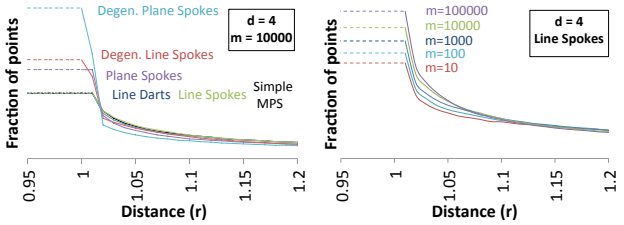
We measure quality via coverage β [Mitchell et al. 2012b; Ebeida et al. 2014a] and the inter-point distance distribution.

Local saturation Figure 4b shows the β guaranteed by the theory ($\beta_{\text{guaranteed}}$) and achieved in practice (β_{achieved}) for different values of m. We see that line-spokes typically achieve a nearly-maximal ($\beta \approx 1$) distribution, and can do so using many fewer spokes than required in theory. Using $\Delta_d = (\beta_{\text{guaranteed}} - 1)/(\beta_{\text{achieved}} - 1)$ for dimension d, we see $\Delta_4 \approx 8$. Figure 4a shows n by m.



(a) *Total inserted points by m.* **(b)** *β for different m*

Figure 4: *Local saturation (coverage) for line-spokes in theory and practice, for the number of spoke misses m. Here $\beta_{guaranteed}$ is the probabilistically-guaranteed saturation upper-bound in theory, and $\beta_{achieved}$ is the β observed in experiments. In practice β is about $8\times$ closer to 1 than the theory guarantee. For example, for $m = 60$ we have $\beta_{achieved} \approx 1.08$, almost maximality, whereas $\beta_{guaranteed} \approx 1.6$. Since $r_f \approx r$ in practice, it is r_c that is determining $\beta = r_c/r_f$.*



(a) *different spoke types* **(b)** *different m*

Figure 5: *Radial profiles, from differential domain distributions [Wei and Wang 2011]. (a) Different sampling types. Spokes all use the same $m = 1000$. Line-spokes, line darts (k-d darts with lines), and Simple MPS produce nearly identical peaks, but line darts is farthest from saturation. (b) Line-spokes with different m. The peak becomes more pronounced as saturation is approached. All distributions are quite flat for larger distances.*

Distribution We analyze sample distributions via Differential Domain Analysis (DDA) [Wei and Wang 2011], which essentially computes histograms of spatial distances between samples. We use DDA instead of Fourier spectral analysis [Lagae and Dutré 2008] because it is faster and easier to compute, especially in high dimensions; and the two are equivalent, differing only by the choice of Gaussian versus sinusoidal kernels [Wei and Wang 2011]. In Figure 5 we plot the 1-d radial means of the high dimensional distributions with respect to different spoke types and m. As shown in Figure 5a, all spoke types exhibit MPS-like characteristics. Degenerate spokes tend to have sharper profiles than full spokes, analogous to the sharper profiles of boundary sampling methods [Dunbar and Humphreys 2006]. Figure 5b shows that higher m will produce sharper profiles due to higher saturation as measured by β.

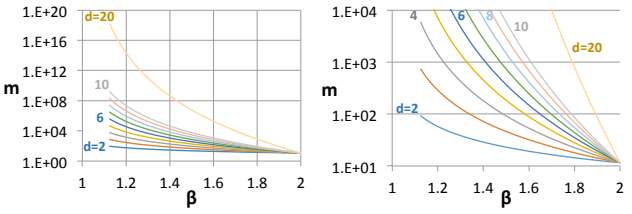
4.3 Parameter Trade-offs

The more spokes we generate, the longer the run-time, but the more saturated the output. Our main control parameter is m, the number of successively-failed spokes for a given extant sample disk before removing it from the front. For line-spokes, our guarantee is that for a given m, with high probability $(1 - \epsilon)$ the achieved $\beta_{achieved}$ is less than β.

How many spoke misses are enough? If the user selects the desired β, then Equation (1) says how large m must be. Conversely, the user may pick m based on a computational budget, and Equation (1) describes what the probabilistically-guaranteed β will be. Note $\beta > 1$, and $-\ln \epsilon > 0$, and $m \geq 1$.

$$m = \left\lceil (-\ln \epsilon)(\beta - 1)^{1-d} \right\rceil \Leftrightarrow \beta = 1 + \left(\frac{-\ln \epsilon}{m} \right)^{1/(d-1)} \quad (1)$$

One can also pick m and β and bound the probability that β was exceeded: $\epsilon < \exp\left(-m(\beta-1)^{d-1}\right)$. In Figure 6, we see that $m = 12$ for $\beta = 2$ and $\epsilon = 10^{-5}$ and all d. The bound on m in Equation (1) is quite practical for moderate dimensions and β.



(a) *full scale* **(b)** *zoom in*

Figure 6: *Illustration of Equation* (1) *with* $\epsilon = 10^{-5}$. *Only 12 successive misses per sample are required to achieve* $\beta = 2$, *regardless of d.*

We provide some proof intuition for Equation (1) here; the actual proofs are in the supplementary material, Appendix A. Let us suppose that the algorithm has terminated and there is a void, some connected part of the domain whose points are farther than r_f from all samples. This void is bounded by some sample disks, and we have thrown at least m spokes from each of these disks. Each of these spokes must have missed this void, otherwise we would have inserted a sample and reset our miss count. The chance of getting m successive misses is the chance of one miss to the mth power, which shows m is dependent on the log of ϵ. The chance of a *single* spoke *missing* this void is the area the void shares with the spoke's disk, divided by the surface area of the disk. Combining this for all disks bounding the void shows that the natural log of the chance that they all missed is (at most) proportional to the area of one disk divided by the total area of the void boundary. Since the void was not hit, the area of the void is probabilistically-guaranteed to be small. A void with small area has a small maximum distance from its interior to its boundary (this is $r_c - r_f$), in particular smaller than the radius of a ball with the same surface area as the void. Thus we get a bound on r_c. The exponential-in-$(d-1)$ dependence on β is precisely the dependence of the surface area of a d-ball on its radius. For $\beta = 2$, we only care about voids with at least the surface area of one of our disks, and this dependence disappears.

In practice, we achieve a much better β than the theoretical guarantee, for all m; See Section 4.2 for a description. This is expected because the proof makes several worst-case assumptions, such as the void being shaped like a ball, and ignores chains of misses less than m.

To bound the overall run-time, we must account for these small miss chains. We assign the cost of a small miss chain to the successful sample disk insertion following it, not the disk generating the chain. Thus each sample accounts for the $(< m-1)$ misses preceding it, the spoke that created it, plus its own m final successive misses, for a total of at most $2m$ spokes. Thus in the entire algorithm we throw at most $2m$ spokes. Each line-spoke takes time $O(dN)$ to trim, where $N < n$ is the number of nearby disks in the pool. Multiplying these gives time $O(dmn^2) = O\left(d(-\ln \epsilon)\,(\beta-1)^{1-d}\, n^2\right)$.

5 Applications

We present applications of our method in Delaunay graph (d = 6–14, Section 5.1), global optimization (d = 6–15, Section 5.2), and motion planning ($d > 20$, Section 5.3). In particular, we use our spoke-dart operations (Section 3.2) for constructing approximate Delaunay graphs from given samples, while global optimization and motion planning can benefit from our placement of new samples in blue noise distributions (Section 3.3). All these applications rely on the underlying domains being sampled as maximally as possible, as measured by β.

5.1 High-dimensional Delaunay Graph

There is an increasing demand for high dimensional meshes in various fields such as uncertainty quantification [Witteveen and Iaccarino 2012] and computational topology [Gerber et al. 2010]. Many applications rely on knowing the distance- and directionally-significant neighbors of points. These applications often rely on Delaunay graphs as a core component. Many methods for constructing the exact Delaunay graph, $\mathcal{D}$, suffer from the curse of dimensionality and their effectiveness deteriorates very rapidly as the dimension increases. Some recent theoretical papers [Miller and Sheehy 2013; Miller et al. 2013] have considered approximate graphs and the problem of dimension from the standpoint of complexity analysis, although no implementations or experimental results of these algorithms are available.

In this section we apply our spoke-dart method to generate an approximate Delaunay graph $\mathcal{D}^*$, which contains with high probability those edges whose dual Voronoi faces subtend a large solid angle with respect to the site vertex. We call these edges *significant Delaunay edges*, and the corresponding $\mathcal{D}^*$ a *significant Delaunay graph*. The significant edges are a subset of the true Delaunay edges, and the Voronoi cell defined by the significant neighbors geometrically contains the true Voronoi cell. Many high dimensional applications, such as the classic approximate nearest neighbor problem, accept approximate Delaunay graphs. One such application is high dimensional global optimization, as shown in Section 5.2. To the best of our knowledge, we present the first practical technique to find a significant Delaunay graph in high dimensions. As a further benefit, for each edge our method produces a *witness*, a domain point on its true Voronoi face, which can be used to estimate the radial extent δ of the Voronoi cell. This is demonstrated in our global optimization application; Section 5.2.

Our basic idea is to throw random line-spokes to tease out the significant Delaunay edges from a set of spatial neighbors. This is a very simple method that scales well across different dimensions. It is summarized in Algorithm 4 with details as follows. We construct the graph $\mathcal{D}^*$ for each vertex s in turn. We initialize its edge pool with all vertices that are close enough to possibly share a Delaunay edge with s. We next identify vertices from this pool who are actual Delaunay neighbors of s with the following probabilistic method. Using spoke-darts, we throw m line-spokes. We trim each spoke ℓ using the separating hyperplane between s and each vertex s' in the pool. There is one pool vertex s^* whose hyperplane trims ℓ the most. (In so-called "degenerate" cases multiple vertices trim the spoke the most and equally. Then we can pick an arbitrary one for s^*.) The far end of the trimmed spoke ω is equidistant from s and s^*, and no other vertex is closer. Hence ω is the witness that s and s^* share a Voronoi-face (Delaunay-edge), and $\overline{ss^*}$ is added to $\mathcal{D}^*$.

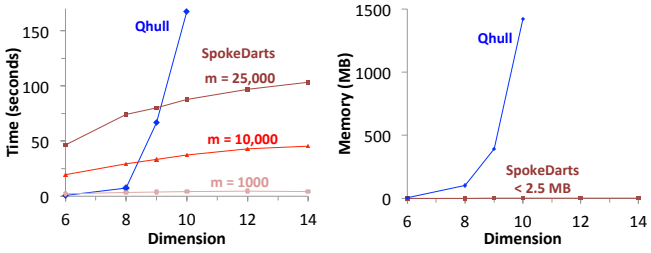
The reason we tend to find the significant neighbors with high probability is obvious from the above algorithm description. Spokes sample the solid angle around each vertex s uniformly, so the probability that a given spoke hits a given Voronoi face is proportional

Input: vertex s, Delaunay graph $\mathcal{D}^*$, NeighborCandidates $\mathcal{M}$, RecursionFlag R
Output: $\mathcal{D}^*$ with s added

$\mathcal{N} = \emptyset$ `// approx. Delaunay neighbors of` s
$\delta(s) = 0$ `// approx. cell radius of` s
for $i = 1$ **to** m **do**
  $\ell \leftarrow$ `RandomLineSpoke`$(s, 0, |\Omega|)$ `// Section 3.2`
  for each sample $s' \in \mathcal{M}$ **do**
    $\pi(s, s') \leftarrow$ hyperplane between s and s'
    trim ℓ with $\pi(s, s')$
    if ℓ got shorter **then**
      $s^* \leftarrow s'$
  $\mathcal{D}^* \leftarrow \mathcal{D}^* \bigcup \{\overline{ss^*}\}$ `// set union without duplication`
  $\mathcal{N} \leftarrow \mathcal{N} \bigcup \{s^*\}$
  $\delta(s) = \max(\delta(s), \texttt{length}(\ell))$
if $R =$ true **then**
  `// update edges of neighbors, removing some`
  for each sample $s' \in \mathcal{N}$ **do**
    $\mathcal{M} \leftarrow$ `Neighbors`$(s') \bigcup \{s\}$
    $\mathcal{D}^* \leftarrow \mathcal{D}^* \setminus$ `Edges`(s') `// remove all edges`
    `Recurse`$(s', \mathcal{D}^*, \mathcal{M}, \text{false})$ `// restore some`
return $\mathcal{D}^*$

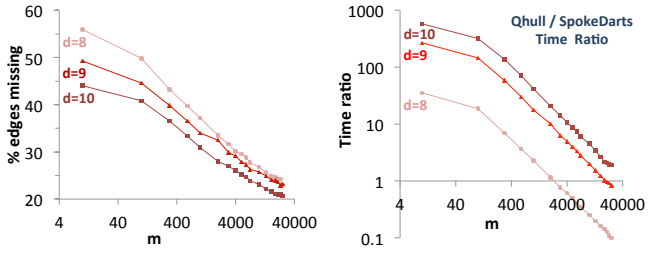
Algorithm 4: *Adding a vertex to the approximate Delaunay graph via our method. For a new vertex $R =$ true.*

to the solid angle the face subtends at s. As the number of spokes m increases we are more likely to also find less significant neighbors, and $\mathcal{D}^* \to \mathcal{D}$. (This is analogous to our blue noise sampling algorithm in Algorithm 3, where larger m increases our chance of finding even the small voids.)



(a) *computation time* **(b)** *memory requirement*

Figure 7: *Comparison of speed (a) and memory (b) between Qhull and spoke-dart sampling for an approximate Delaunay graph. Qhull becomes infeasible beyond $d = 10$ whereas our method scales well.*



(a) *effects of m on % of missing edges* **(b)** *effects of m on time*

Figure 8: *Effects of m on the approximate Delaunay graph. As m increases, fewer Delaunay edges are missed (a) but run-time increases (b).*

We demonstrate the efficiency of our approach against Qhull [Barber et al. 1996], a commonly-used code for convex hulls and Delaunay triangulations. As test input, we used Poisson-disk point sets over the unit-box domain in various dimensions. For each case, we used Qhull to generate the exact solution $\mathcal{D}$ and our method for the approximate solution $\mathcal{D}^*$. As Figure 7 shows, the memory and time requirements of Qhull grows significantly as d increases. Qhull required memory that might not be practical for $d \geq 11$. On the other hand, our method shows a linear growth for time and memory with d. We see that our method became competitive for $d \geq 9$. Figure 8 shows the effect of m on the time and number of missed edges.

5.2 Rethinking Lipschitzian Optimization

A variety of disciplines — science, engineering or even economics, — seek the "absolutely best" answer. This usually involves solving a global optimization problem, where one explores a high-dimensional design space to find the optimum of some objective function under a set of feasibility constraints. Local optimality is not enough. For simple analytical functions, some algorithms are guaranteed to find the global minimum. However, no method is guaranteed to find the global minimum for all functions, or even come close in finite time; for example, no method is guaranteed to find the minimum of a function resembling white noise. Heuristic stochastic techniques are usually the best in practice, and sometimes the only option [Horst et al. 2002]. In particular, we consider the important category of Lipschitzian optimization methods for complex but well-behaved, high-dimensional functions. We demonstrate how spoke-dart sampling can improve upon DIRECT, which for many decades has been a preferred method. We believe this opens the door to new approaches.

Lipschitzian optimization [Shubert 1972] explores the parameter space and provides convergence based on the Lipschitz constant of the objective function. Specifically, a function f is *Lipschitz continuous* with constant $K > 0$ if

$$|f(x_i) - f(x_j)| \leq K|x_i - x_j| \tag{2}$$

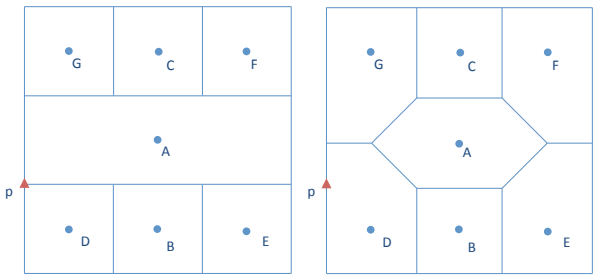
for all $x_i \neq x_j$ in the feasible domain of f. One can use this condition to show that a neighborhood around a sample point cannot contain the best solution, and hence can be discarded. In particular, if K is known and the best currently-known answer is f^*, then a ball around x_i of radius $|f(x_i) - f^*|/K$ has values above f^*. We only need to search the space outside this ball.

DIRECT [Shubert 1972] has two limitations: poor scaling to high dimensions; and relying on a global K, whose exact value is often unknown. The DIRECT algorithm [Jones et al. 1993] generalizes [Shubert 1972] to higher dimensions and does not require knowledge of the Lipschitz constant. DIRECT partitions the domain into hyperrectangles. It refines those rectangles that could contain a better point than the currently best-known f^*. This refinement recurses until reaching the maximum number of iterations, or the remaining possible improvement is small. In particular, DIRECT deterministically decides to refine the jth rectangle if

$$f(c_j) - \tilde{K}\delta_j \leq f(c_i) - \tilde{K}\delta_i \quad \forall\, i = 1, 2, ..., m, \text{ and} \tag{3}$$

$$f(c_j) - \tilde{K}\delta_j \leq f^* - \epsilon\,|f^*|\,. \tag{4}$$

Here c_j is the center of the rectangle, and δ_j is the distance from the rectangle's center to its (farthest) corner. Also $\tilde{K}$ runs over all positive real numbers, and the index i runs through all cells in the domain. The best currently-known value is f^*, and ϵ is a small positive number. Intuitively, DIRECT avoids the need to know K



(a) *rectangular cells* **(b)** *Voronoi cells*

Figure 9: *Main weakness of DIRECT [Jones et al. 1993]. Rectangular partitions (a) can give misleading estimates of sample-neighborhood sizes. Voronoi cell partitions (b) improve these estimates, especially for high dimensions. For example, the size of the relevant neighborhood of point A in (a) is overestimated, since all its corners are actually closer to other sample points, e.g. p is closer to D, as seen in (b).*

via Equation (3), in which we consider whether any $\tilde{K}$ could allow the cell to contain the global optimum. For small values of $\tilde{K}$, Equation (4) avoids selecting cells which can lead only to minor improvements. The set $\mathcal{L}$ of all rectangles satisfying these equations, including their $\tilde{K}$ values, can be computed efficiently via the convex hull [Jones et al. 1993]. Specifically, $\mathcal{L}$ is the lower envelope of the convex-hull of points (representing rectangles) plotted in the two-dimensional domain δ by f. Since DIRECT uses rectangles, many cells have the same δ and the data points tend to stack above one another.

However, we question the efficacy of rectangular cells in DIRECT. As illustrated in Figure 9, they do not appear to be the best way to describe or measure local neighborhoods around sample points. Rectangles give misleading δ_i, and slow DIRECT's convergence rate, especially in high dimensions.

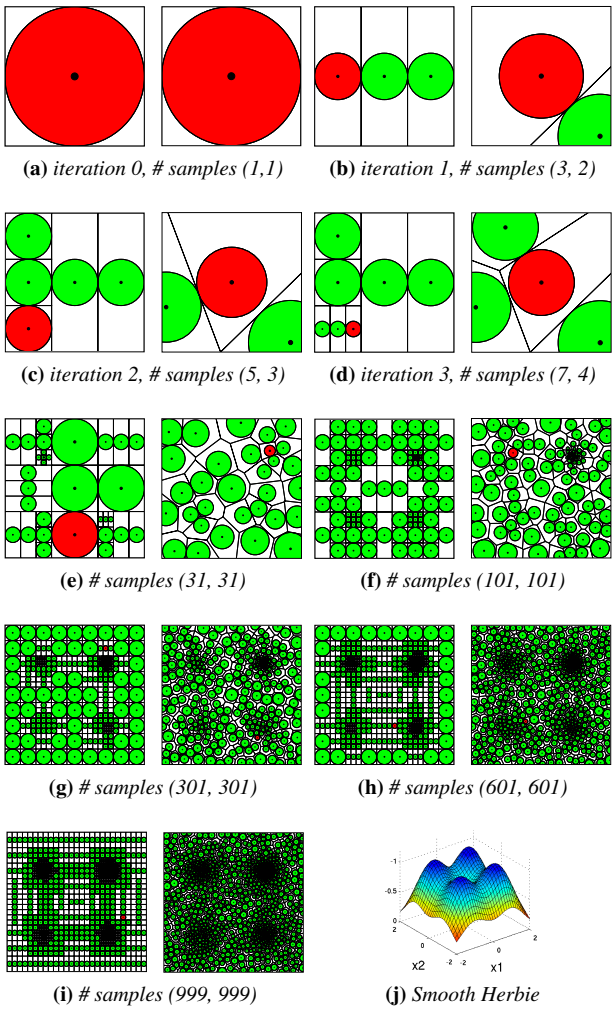
Input: target function f over domain Ω
Output: minimum f^* found

$s = \texttt{Center}(\Omega)$ `// any sample point`
$\mathcal{S} = \{s\}$ `// sample set`
$\mathcal{F} \leftarrow f(s)$ `// function values at samples`
$f^* \leftarrow f(s)$ `// minimum value found so far`
$\{\delta\} = \{|\Omega|\}$ `// set of cell size estimates`
while computational budget is not exhausted **do**
  $\mathcal{L} = \texttt{LowerHull}(\mathcal{F}, \{\delta\}, f^*)$
  $s \leftarrow \texttt{RandomSelect}(\mathcal{L})$
  $s' \leftarrow \texttt{OneSpokeSample}(s, \mathcal{D}^*(s))$
  $\mathcal{S} \leftarrow \mathcal{S} \bigcup \{s'\}$
  $\mathcal{F} \leftarrow \mathcal{F} \bigcup \{f(s')\}$
  $f^* \leftarrow \min(f^*, f(s'))$
  $(\mathcal{D}^*, \{\delta\}) \leftarrow \texttt{DelaunayAdd}(\mathcal{D}^*, s')$ `// Algorithm 4`
return f^*

Algorithm 5: *Lipschitzian optimization via our method.*

Our method We follow the basic steps in DIRECT but improve it in two major aspects: using Voronoi regions instead of hyper-rectangular cells, and placing samples via stochastic blue noise instead of deterministic cell division. In particular, to refine a cell, we first add a new sample within it via our spoke-dart sampling algorithm. We set the conflict radius to the cell's inscribed hypersphere radius, to avoid adding a sample point that is too close to a prior sample. We then divide the cell (and update its neighboring cells) via the approximate Delaunay graph as described in Section 5.1, and use the computed witnesses to estimate the δ values in Equation (3). These two steps replace the corresponding deterministic center-sample and rectangular cell division in DIRECT, respectively. See Algorithm 5 for a summary.

To our knowledge, our method is the first exact stochastic Lipschitzian optimization technique that combines the benefits of guaranteed convergence in [Jones et al. 1993] and high dimensional efficiency in [Spall 2005]. Computing blue noise and Voronoi regions has been intractable in high dimensions, and this is probably why this direction has not been explored before.



(a) *iteration 0, # samples (1,1)* **(b)** *iteration 1, # samples (3, 2)*
(c) *iteration 2, # samples (5, 3)* **(d)** *iteration 3, # samples (7, 4)*
(e) *# samples (31, 31)* **(f)** *# samples (101, 101)*
(g) *# samples (301, 301)* **(h)** *# samples (601, 601)*
(i) *# samples (999, 999)* **(j)** *Smooth Herbie*

Figure 10: *Comparing DIRECT (left) and our method (right) while exploring the smooth Herbie function (j). We list the number of samples used by each: (DIRECT, us).*

Demonstration A didactic comparison of DIRECT and our method is illustrated in Figure 10. It uses the smooth Herbie function, a 2-d test function popular in the optimization community because it has four local optima of similar value, located in different quadrants. Notice how DIRECT partitions the space via deterministic rectangles while our method uses blue-noise Voronoi cells. Table 1 shows the superior performance of our method over a set of benchmark high-dimensional functions, where the standard measure of performance is the number of function evaluations needed,

Benchmark	dimension	DIRECT	Our method	Speedup
Easom	6	4987	1912	2.60×
Easom	8	64405	8480	7.59×
Easom	10	816937	19081	42.81×
Bohachevsky	7	10315	2125	4.85 ×
Exponential	10	13481	7807	1.72 ×
Exponential	15	36890	10316	3.57×

Table 1: *Performance comparison of DIRECT and our method, measured by the number of function evaluations needed to find the global minimum within relative error* $\epsilon = \dfrac{f^* - f_{min}}{f_{max} - f_{min}} = 10^{-4}$. *Since our method is random, results are the averages over 100 runs.*

because in real problems those tend to be very expensive and dominate the overall cost.

5.3 High-dimensional Motion Planning

Motion planning algorithms are frequently used in robotics, gaming, CAD/CAM, and animation [Yamane et al. 2004; Overmars 2005; Pan et al. 2010]. The main goal is to compute a collision-free path for real or virtual robots among obstacles. Furthermore, the resulting path may need to satisfy additional constraints, including path smoothness, dynamics constraints, and plausible motion for gaming or animation. This problem has been extensively studied in many areas for more than three decades. Two main challenges are:

Speed The computation needs to be fast enough for interactive applications and dynamic environments.

Dimensionality High Degrees-Of-Freedom (DOF) robots are very common. For example, the simplest models for humans (or humanoid robots) have tens of DOF, capable of motions like walking, sitting, bending or picking objects.

Some of the most popular algorithms for high-DOF robots use sample-based planning [LaValle and Kuffner 2001]. The main idea is to generate random collision-free sample points in the high-dimensional configuration space, and join the nearby points using local collision-free paths. Connected paths provide a roadmap or tree for path computation or navigation. In particular, RRT (Rapidly-exploring Random Tree) [Kuffner and Lavalle 2000] incrementally builds a tree from the initial point towards the goal configuration. RRT is relatively simple to implement and widely used in many applications.

However, prior RRT methods generate samples via white noise (a.k.a. Poisson process). These samples are not uniformly spaced in the configuration space, leading to suboptimal computation. Using Poisson-disk sampling instead can lead to more efficient exploration of the configuration space, as demonstrated in a recent work by Park et al. [2013]. As summarized in Algorithm 6, the method uses a precomputed Poisson-disk sampling to guide the generation of new points which are not too close to prior points. Adaptive sampling can also be used to generate more samples in tight spaces. The performance of RRT planning can be further improved by multi-core GPUs.

Due to the curse-of-dimensionality, Park et al. [2013] has been restricted to relatively low dimensional spaces, $d \leq 6$. Our method offers help here by simply precomputing the sample set via spoke-dart sampling. We use three well-known motion planning benchmark scenarios from OMPL [Şucan et al. 2012] to evaluate the performance of the planning algorithm. These scenarios all have 6 DOF, and vary in their level of difficulty. We also compute the motion of the HRP-4 robot with 23 DOF; see Figure 1c. The total times

Input: start / goal configurations $\mathbf{x}_{init}$ and $\mathbf{x}_{goal}$ within domain Ω
Input: Poisson-disk sample set $\mathbf{P}$ precomputed via Algorithm 3
Output: RRT Tree $\mathbf{T}$

$\mathbf{T}$.add($\mathbf{x}_{init}$)
$\mathbf{P}$.add($\mathbf{x}_{goal}$)
for $i = 1$ **to** m **do in parallel** `// multiple threads`
  while $\mathbf{x}_{goal} \notin \mathbf{T}$ **do**
    $\mathbf{y} \leftarrow$ RandomSample(Ω)
    $\mathbf{T} \leftarrow$ Extend($\mathbf{T}, \mathbf{y}, \mathbf{P}$)
end for
return $\mathbf{T}$

Algorithm 6: *Parallel Poisson-RRT with precomputed samples.*

Benchmark	DOF	RRT (1 CPU core)	GPU Poisson-RRT	Speed-up
Easy	6	0.34	0.03	12.14×
AlphaPuzzle	6	32.76	1.31	24.93×
Apartment	6	191.79	11.88	16.15×
HRP-4	23	6.17	0.32	19.28×

Table 2: *Comparison of the performances of our GPU-based Poisson-RRT planning algorithms and a reference single-core CPU algorithm. We compared the planning time for different benchmarks using 100 trials.*

taken by the *planner* are shown in Table 2. For *sampling* time, the only competition comes from line darts, and we have demonstrated in Figure 3 that our spoke-dart sampling is more efficient.

6 Conclusions and Future Work

In summary, we have presented spoke-dart sampling as a new algorithm for generating well-spaced blue noise distributions in high dimensions. The method combines the advantages of state-of-the-art methods: the locality of advancing-front and simplicity of k-d darts. We demonstrated the usefulness of our method for a variety of applications.

Our method has several parameters. Usually the user has no choice over the domain dimension d. If quick run-time is desired, then select $m = 12$ consecutive misses. If higher saturation ($\beta < 2$) is desired, use Equation (1) to select m, but be prepared to wait in high dimensions. In any event, memory should be a minor issue. Degenerate spokes are faster than full spokes in terms of the number of points inserted; this advantage tends to disappear as the dimension increases beyond six. Plane-spokes are faster than line-spokes; since they effectively reduce the dimension by (only) one, this advantage tends to disappear as the dimension increases. Moreover, both degenerate spokes and plane-spokes may be producing more points more quickly merely because they are inserting more points at distance r_c and creating a tighter, less-random packing than maximal Poisson-disk sampling. As such, there is little to recommend degenerate spokes. Our overall recommendation is to use full line-spokes. We use line-spokes of extent twice the Poisson-disk radius, and select the next sample uniformly from the nearest segment. We would like to understand the effect of the extent and the selection criteria on the final output distribution.

We would like to analyze and improve the accuracy of generating approximate Delaunay graphs. We speculate that our approximate Delaunay graphs may supplant the use of k-nearest neighbors for computational topology and manifold learning. The benefit is that our Delaunay graph considers all directions; in contrast, for a point near a dense cluster, k-nearest neighbors can miss significant neighbors in directions opposite to the cluster.

Spoke-darts may inspire further research in global optimization.

We presented the approach and demonstrated it on a small set of benchmarks. In our current implementation for motion planning we precompute all samples. We are investigating the possibility of sampling on the fly by exploiting the similarity between our method and RRT tree growth. A potential application for high dimensional blue noise sampling is rendering. Beyond sampling, we believe spoke-darts can also benefit numerical integration as demonstrated in Ebeida et al. [2014b].

References

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LÉVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. In *SIGGRAPH '03*, 485–493.

ARYA, S., AND MOUNT, D. M. 2005. *The Handbook of Data Structures and Applications*. Chapman & Hall/CRC, Boca Raton, ch. Computational Geometry: Proximity and Location, 63.1–63.22. eds. D. Mehta and S. Sahni.

BALZER, M., SCHLOMER, T., AND DEUSSEN, O. 2009. Capacity-constrained point distributions: A variant of Lloyd's method. In *SIGGRAPH '09*, 86:1–8.

BARBER, C. B., DOBKIN, D. P., AND HUHDANPAA, H. 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS) 22*, 4, 469–483.

BRIDSON, R. 2007. Fast Poisson disk sampling in arbitrary dimensions. In *SIGGRAPH '07: ACM SIGGRAPH 2007 Sketches & Applications*.

CHEN, J., GE, X., WEI, L.-Y., WANG, B., WANG, Y., WANG, H., FEI, Y., QIAN, K.-L., YONG, J.-H., AND WANG, W. 2013. Bilateral blue noise sampling. *ACM Trans. Graph. 32*, 6 (Nov.), 216:1–216:11.

CLINE, D., JESCHKE, S., RAZDAN, A., WHITE, K., AND WONKA, P. 2009. Dart throwing on surfaces. In *EGSR '09*, 1217–1226.

COOK, R. L. 1986. Stochastic sampling in computer graphics. *ACM Trans. Graph. 5*, 1, 51–72.

ŞUCAN, I. A., MOLL, M., AND KAVRAKI, L. E. 2012. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine 19*, 4, 72–82. `http://ompl.kavrakilab.org`.

DE GOES, F., BREEDEN, K., OSTROMOUKHOV, V., AND DESBRUN, M. 2012. Blue noise through optimal transport. In *SIGGRAPH Asia '12*, 171:1–171:11.

DIPPÉ, M. A. Z., AND WOLD, E. H. 1985. Antialiasing through stochastic sampling. In *SIGGRAPH '85*, 69–78.

DUNBAR, D., AND HUMPHREYS, G. 2006. A spatial data structure for fast Poisson-disk sample generation. In *SIGGRAPH '06*, 503–508.

EBEIDA, M. S., PATNEY, A., MITCHELL, S. A., DAVIDSON, A., KNUPP, P. M., AND OWENS, J. D. 2011. Efficient maximal Poisson-disk sampling. In *SIGGRAPH '11*, 49:1–12.

EBEIDA, M. S., MITCHELL, S. A., PATNEY, A., DAVIDSON, A. A., AND OWENS, J. D. 2012. A simple algorithm for maximal Poisson-disk sampling in high dimensions. *Comp. Graph. Forum 31*, 2pt4, 785–794.

EBEIDA, M. S., MAHMOUD, A. H., AWAD, M. A., MOHAMMED A. MOHAMMED, S. A. M. A. R., AND OWENS, J. D. 2013. Sifted disks. *Comp. Graph. Forum 32*, 2.

EBEIDA, M. S., AWAD, M. A., GE, X., MAHMOUD, A. H., MITCHELL, S. A., KNUPP, P. M., AND WEI, L.-Y. 2014. Improving spatial coverage while preserving blue noise of point sets. *Computer-Aided Design 46* (January), 25–36.

EBEIDA, M. S., PATNEY, A., MITCHELL, S. A., DALBEY, K. R., DAVIDSON, A. A., AND OWENS, J. D. 2014. k-d darts: Sampling by k-dimensional flat searches. *ACM Trans. Graph. 33*, 1 (Feb.), 3:1–3:16.

FATTAL, R. 2011. Blue-noise point sampling using kernel density model. In *SIGGRAPH '11*, 48:1–12.

GAMITO, M. N., AND MADDOCK, S. C. 2009. Accurate multidimensional Poisson-disk sampling. *ACM Trans. Graph. 29*, 1, 1–19.

GERBER, S., BREMER, P., PASCUCCI, V., AND WHITAKER, R. 2010. Visual exploration of high dimensional scalar functions. *Visualization and Computer Graphics, IEEE Transactions on 16*, 6, 1271–1280.

HORST, R., PARDALOS, P. M., AND ROMEIJN, H. E. 2002. *Handbook of global optimization*, vol. 2. Springer.

JONES, D. R., PERTTUNEN, C. D., AND STUCKMAN, B. E. 1993. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications 79*, 1, 157–181.

KUFFNER, J. J., AND LAVALLE, S. M. 2000. RRT-Connect: An efficient approach to single-query path planning. In *Proc. IEEE Conf. on Robotics and Automation*, 995–1001.

LAGAE, A., AND DUTRÉ, P. 2008. A comparison of methods for generating Poisson disk distributions. *Computer Graphics Forum 21*, 1, 114–129.

LAVALLE, S., AND KUFFNER, J. 2001. Randomized kinodynamic planning. *International Journal of Robotics Research 20*, 5, 378–400.

LI, X.-Y., TENG, S.-H., AND ÜNGÖR, A. 1999. Biting: Advancing front meets sphere packing. In *Int. Jour. for Numerical Methods in Eng*.

LI, H., WEI, L.-Y., SANDER, P., AND FU, C.-W. 2010. Anisotropic blue noise sampling. In *SIGGRAPH Asia '10*, 167:1–12.

LIU, J., LI, S., AND CHEN, Y. 2008. A fast and practical method to pack spheres for mesh generation. *Acta Mechanica Sinica 24*, 4, 439–447.

LIU, J. 1991. Automatic triangulation of N-dimensional Euclidean domains. In *Proceedings of CAD/Graphics '91*, 238–241.

MILLER, G. L., AND SHEEHY, D. R. 2013. A new approach to output-sensitive Voronoi diagrams and Delaunay triangulations. In *SOCG: Proceedings of the 29th ACM Symposium on Computational Geometry*.

MILLER, G. L., SHEEHY, D. R., AND VELINGKER, A. 2013. A fast algorithm for well-spaced points and approximate Delaunay graphs. In *SOCG: Proceedings of the 29th ACM Symposium on Computational Geometry*.

MITCHELL, S. A., RAND, A., EBEIDA, M. S., AND BAJAJ, C. 2012. Variable radii Poisson-disk sampling, extended version. In *Proceedings of the 24th Canadian Conference on Computational Geometry*, 1–9.

MITCHELL, S. A., RAND, A., EBEIDA, M. S., AND BAJAJ, C. 2012. Variable radii Poisson-disk sampling. In *Proceedings of the 24th Canadian Conference on Computational Geometry*, 185–190.

MULLER, M. E. 1959. A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM 2*, 4 (Apr.), 19–20.

OVERMARS, M. H. 2005. Path planning for games. In *Proc. 3rd*

Int. Game Design and Technology Workshop, 29–33.

ÖZTIRELI, A. C., ALEXA, M., AND GROSS, M. 2010. Spectral sampling of manifolds. In *SIGGRAPH ASIA '10*, 168:1–8.

PAN, J., ZHANG, L., LIN, M. C., AND MANOCHA, D. 2010. A hybrid approach for simulating human motion in constrained environments. *Computer Animation and Virtual Worlds 21*, 3-4, 137–149.

PARK, C., PAN, J., AND MANOCHA, D. 2013. Real-time optimization-based planning in dynamic environments using gpus. In *ICRA '13*, 4090–4097.

SAMET, H. 2005. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc.

SCHECHTER, H., AND BRIDSON, R. 2012. Ghost SPH for animating water. *ACM Trans. Graph. 31*, 4, 61:1–61:8.

SHUBERT, B. O. 1972. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis 9*, 3, 379–388.

SPALL, J. C. 2005. *Introduction to stochastic search and optimization: estimation, simulation, and control*, vol. 65. Wiley. com.

SUN, X., ZHOU, K., GUO, J., XIE, G., PAN, J., WANG, W., AND GUO, B. 2013. Line segment sampling with blue-noise properties. In *SIGGRAPH '13*.

WEI, L.-Y., AND WANG, R. 2011. Differential domain analysis for non-uniform sampling. In *SIGGRAPH '11*, 50:1–10.

WITTEVEEN, J. A., AND IACCARINO, G. 2012. Simplex stochastic collocation with random sampling and extrapolation for nonhypercube probability spaces. *SIAM Journal on Scientific Computing 34*, 2, A814–A838.

YAMANE, K., KUFFNER, J. J., AND HODGINS, J. K. 2004. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics (TOG) 23*, 3, 532–539.

YAN, D.-M., AND WONKA, P. 2013. Gap processing for adaptive maximal poisson-disk sampling. *ACM Trans. Graph. 32*, 5 (Oct.), 148:1–148:15.

A Bound Proofs for Section 4.3

Here we provide bounds on m, β, and ϵ in terms of d. We consider line-spokes only. A *void* is an uncovered region. It is bounded by some disks. The chance of hitting a void will depend on its *surface area* Area(void), the d-1 dimensional volume of its boundary.

A.1 Chance of missing the void from one disk

Let us quantify the chance $p_1(\text{miss})$ that a line-spoke from disk D_1 missed a void. See Figure 11a. Let $R_1 = \text{Area}(\text{void} \cap D_1)/\text{Area}(D_1)$. Since line-spokes are chosen uniformly from the surface area of the disk, $p_1(\text{hit}) = R_1$, and $p_1(\text{miss}) = 1 - R_1$. (We may multiply the hit chance by 2 if the extent of the void is small enough that it does not contain antipodal points of the disk and we use a line rather than a ray for a spoke.)

The chance of missing the void consecutively m times is then $p_1^m(\text{miss}) = \prod_{j=1}^{m}(1 - R_1) = (1 - R_1)^m$. Using the well-known inequality $e^{-x} = \exp(-x) > 1 - x$, we have $p_1^m(\text{miss}) < \exp(-mR_1)$.

A.2 Chance of missing the void from all disks

The chance of missing m times consecutively from all N bounding disks is then $p_{\text{all}}^m(\text{miss}) = \prod_{i=1}^{N} p_i^m(\text{miss}) < \exp\left(-m\sum_{i=1}^{N} R_i\right) = \exp(-mR)$, where all sample disks have the same radius and size so we can drop their subscripts and $R = \text{Area}(\text{void})/\text{Area}(D)$.

If we wish this miss chance to be less than ϵ, then it is sufficient to have $\exp(-mR) < \epsilon$ or $mR > -\ln(\epsilon) > 0$.

A.3 Bound in terms of β

Now we bound R in terms of β. Suppose there is a domain point v in the void at distance r_c from all samples. Then a ball at v of radius $r_{\text{void}} = r_c - r_f$ is strictly inside the void, and $\text{Area}(\text{void}) > \text{Area}(D(r_{\text{void}}))$; see Figure 11b. Since we are in d dimensions and $\beta = r_c/r_f$,

$$R = \frac{\text{Area}(\text{void})}{\text{Area}(D)} > \frac{r_{\text{void}}^{d-1}}{r_f^{d-1}} = (\beta - 1)^{d-1}$$

Hence a sufficient condition is $m(\beta - 1)^{d-1} > -\ln\epsilon$, or

$$m = \left\lceil (-\ln\epsilon)(\beta - 1)^{1-d} \right\rceil \Leftrightarrow \beta = 1 + \left(\frac{-\ln\epsilon}{m}\right)^{1/(d-1)} \quad (5)$$

A.4 Example m Values

Table 3 gives example m values using Equation (5).

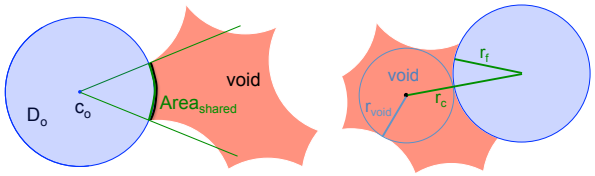
A.5 Other issues

Boundary caveats The astute reader may have noticed that we made no mention of the domain boundary. The void disk $D(r_{\text{void}})$ must be inside the domain, and $\text{Area}(\text{void}) > \text{Area}(D(r_{\text{void}}))$ is only guaranteed to hold for non-periodic domains. Here we assumed that the void was bounded by disks only, and not the domain boundary. For bounded domains, this may be finessed by throwing spokes on or near the domain boundary to ensure it is covered.

d \ β	2	1.5	1.25	1.125
2	12	24	47	93
3	12	47	185	737
4	12	93	737	5900
5	12	185	3.0e3	4.8e4
6	12	369	1.2e4	3.8e5
7	12	737	4.8e4	3.1e6
8	12	1.5e3	1.9e5	2.5e7
9	12	3.0e3	7.6e5	2.0e8
10	12	5.9e3	3.1e6	1.6e9
20	12	6.1e6	3.2e12	1.7e18
30	12	6.2e9	3.4e18	1.8e27
40	12	6.4e12	3.5e24	2.0e36
50	12	6.5e15	3.7e30	2.1e45
100	12	3.4e21	4.7e60	3.0e90

Table 3: *Values of m by d and β for $\epsilon = 10^{-5}$ as computed from Equation* (5).

Order independence There is a statistical subtly in Appendix A.2. It does not matter that the consecutive spokes from one bounding disk were not consecutive with the spokes from another disk. The misses for each of the remaining boundary pieces is independent of whether the void was hit and reduced by some spokes from a later front disk. The important thing is that no spoke ever hit the boundary of the terminal, remaining void.



(a) *Shared area of a void and a disk.* **(b)** *An empty ball in a void has smaller surface area than the void.*

Figure 11: *Hitting a void from a neighboring disk.*