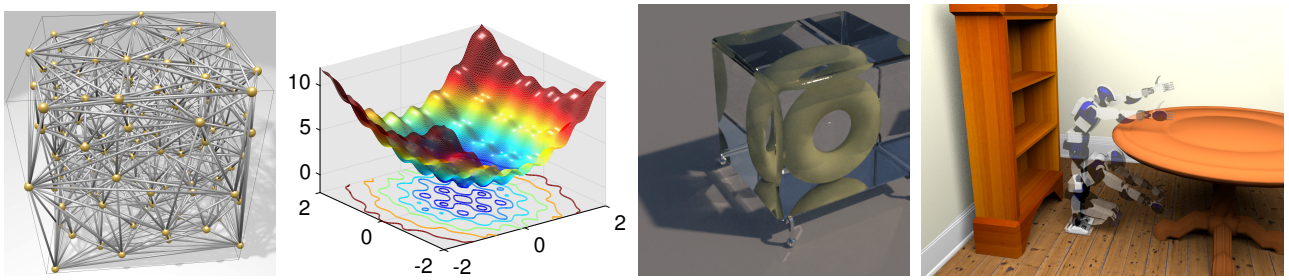
Spoke-Darts for High-Dimensional Blue Noise Sampling

SCOTT A. MITCHELL, Sandia National Laboratories
MOHAMED S. EBEIDA, Sandia National Laboratories
MUHAMMAD A. AWAD, University of California at Davis
CHONHYON PARK, UNC Chapel Hill
ANJUL PATNEY, NVIDIA Research
AHMAD A. RUSHDI, University of California at Davis and Sandia National Laboratories
LAURA P. SWILER, Sandia National Laboratories
DINESH MANOCHA, UNC Chapel Hill
LI-YI WEI, University of Hong Kong and Adobe Research



(a) 8D Delaunay graph (b) 100D global optimization (c) 8D rendering (d) 23D motion planning

Fig. 1. Spoke-dart sampling for high-dimensional applications: Delaunay graph construction, optimization, rendering, and motion planning.

Blue noise sampling has proved useful for many graphics applications, but remains under-explored in high-dimensional spaces due to the difficulty of generating distributions and proving properties about them. We present a blue noise sampling method with good quality and performance across different dimensions. The method, spoke-dart sampling, shoots rays from prior samples and selects samples from these rays. It combines the advantages of two major high-dimensional sampling methods: the locality of advancing front with the dimensionality-reduction of hyperplanes, specifically line sampling. We prove that the output sampling is saturated with high probability, with bounds on distances between pairs of samples, and between any domain point and its nearest sample. We demonstrate spoke-dart applications for approximate Delaunay graph construction, global optimization, and robotic motion planning. Both the blue-noise quality of the output distribution, and the adaptability of the intermediate processes of our method, are useful in these applications.

CCS Concepts: • **Computing methodologies** → *Antialiasing*;

Additional Key Words and Phrases: line sampling, high dimension, blue noise, Delaunay graph, global optimization, motion planning

ACM Reference format:
Scott A. Mitchell, Mohamed S. Ebeida, Muhammad A. Awad, Chonhyon Park, Anjul Patney, Ahmad A. Rushdi, Laura P. Swiler, Dinesh Manocha, and Li-Yi Wei. 2018. Spoke-Darts for High-Dimensional Blue Noise Sampling. *ACM Trans. Graph.* 37, 2, Article 22 (May 2018), 20 pages.

© 2018 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery. Author email: *samitch@sandia.gov.*
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, https://doi.org/10.1145/3194657.

https://doi.org/10.1145/3194657

1 INTRODUCTION

Sampling is a core technique for various scientific and engineering applications. Sampling allows us to approximate continuous quantities in tractable space and time via discrete samples. The samples should be well-spaced for efficiency, and yet random enough to avoid structural aliasing. Low-discrepancy sequences [Keller et al. 2012; Niederreiter 1992] are known for generating well-spaced samples, but their inherently deterministic nature is prone to produce regular patterns, which can cause aliasing. Blue noise sampling [Ebeida et al. 2012; Lagae and Dutré 2008; Mitchell 1987; Ulichney 1988] can synthesize samples that are simultaneously random and well-spaced, but it is computationally more expensive, especially as the dimension increases. Hyperplane sampling [Ebeida et al. 2014] is a random approach that scales to any dimension, but the output distribution is not guaranteed to be well-spaced. Thus, high-dimensional (e.g. $\geq$ 6D) blue-noise sampling remains elusive, even though high-dimensional spaces are common in geometry, optimization, rendering, robotics and other applications.

Advancing front techniques [Bridson 2007; Dunbar and Humphreys 2006; Li et al. 2000; Liu 1991; Liu et al. 2008] are able to efficiently sample from irregular domains; in contrast many other methods are tailored to domains that are hyperrectangles. The ability to handle irregular domains is an advantage in some contexts, but the complexity of computing and storing the geometry of fronts grows exponentially with dimension.

We present a new algorithm that has the advantages of both advancing front and hyperplane sampling. It scales to high dimensions by avoiding computing the front geometry. It uses line sampling, selecting the next sample from a line segment through a prior sample.

Its output has guaranteed blue noise properties. We provide bounds on the spatial properties of our output, including saturation, that apply to any dimension. We show algorithmic time and space complexities that avoid the curse of dimensionality. We provide experiments that confirm these theoretical bounds and trends, and compare to related methods.

Traditional blue noise does well at avoiding low frequency artifacts. To avoid artifacts in high frequency areas, the community has developed an interest in *step blue noise* [Heck et al. 2013], where the frequency spectrum resembles a step function without oscillations. Until now, the only way to create these distributions was an expensive post-processing optimization of an initial distribution. Variations of our algorithm can create *soft blue noise*, potentially avoiding high and low frequency artifacts.

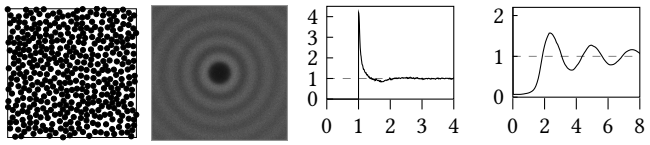
Beyond blue noise, the adaptability and efficiency of our methods facilitate diverse applications, as shown in Figure 1. For *approximate Delaunay graph construction* and *global optimization*, the advancing-front and radial exploration provide advantages when sampling from the irregular shape of the local domains, even when the global domain is a hyper-rectangle. Moreover, global optimization benefits from the intermediate process of our sampling method, not just the final output sample sets. For *robotic motion planning*, the ability to do advancing front over irregular domains may prove useful for adaptively exploring narrow regions of the configuration space.

The contributions of this paper include:

- The idea of *spoke-dart sampling*, which combines the advantages of the locality of advancing-front with the dimension-mitigation of hyperplane sampling, specifically line-sampling;
- Direct algorithms for blue noise in high dimensions;
- Proven and demonstrated time, memory, and saturation bounds that scale well;
- Applications using spokes for high dimensional Delaunay graphs, global optimization, and motion planning;
- Open source software [Awad et al. 2016].

To our knowledge, we provide the first method for probabilistically-guaranteed locally-saturated blue noise in high dimensions, and the first direct method for *soft blue noise* in $d > 2$. By "direct" we mean that samples are placed once, when they are generated, and never moved. We demonstrate blue noise in dimensions 2–30, and applications in dimensions up to 100.

2 RELATED WORK



(a) Point set (b) Periodogram (c) RDF (d) RP

Fig. 2. Randomness metrics from PSA [Schlömer and Deussen 2011].

Blue noise has many graphics applications [Chen et al. 2013] in rendering [Cook 1986; Sun et al. 2013], texturing [Lagae and Dutré 2005], stippling [Balzer et al. 2009; de Goes et al. 2012; Fattal 2011], geometry processing [Alliez et al. 2003; Öztireli et al. 2010], animation [Schechter and Bridson 2012], visualization [Li et al. 2010], and numerical computation [Ebeida et al. 2014]. Blue noise sampling can be achieved by various methods, such as dart throwing (also known as Poisson-disk sampling) [Cook 1986] and relaxation [Lloyd 1983].

Two main spatial properties are used to characterize blue noise distributions: (1) randomness and (2) well-spaced-ness. Randomness avoids aliasing while well-spaced-ness reduces noise and improves efficiency.

Randomness is typically characterized by the frequency spectrum of the sample distribution [Ulichney 1988], a feature of the output of some process rather than the randomness of the process itself. The spectral properties can be measured by the radial power (RP) [Lagae and Dutré 2008] in the frequency domain, or equivalently in the primary domain such as differential vectors (differential domain analysis) [Wei and Wang 2011] and radial distance function (RDF) [Öztireli and Gross 2012]. See Figure 2 for an example of these measures. Many traditional algorithms for blue noise produce a step-like RDF, but a RP spectrum with oscillations, and these can produce visible artifacts in high frequency areas [Heck et al. 2013]. Step blue noise has a RP resembling a step function, without oscillations. Stair blue noise [Kailkhura et al. 2016] provides additional degrees of freedom over step blue noise for tuning spectral characteristics. By "soft blue noise" we loosely mean that both the RP *and* RDF are steep but smooth ramps without oscillations. This can be preferred because of lower aliasing, as demonstrated by recent results of using this type of noise for the classical zone-plate pattern [Heck et al. 2013; Kopf et al. 2006; Subr and Kautz 2013]. Our prior two-radii sampling directly produced soft blue noise in 2D, without post-processing [Mitchell et al. 2012].

Well-spaced samples, on the other hand, mean that samples are not too close to one another, yet no domain point is too far from a sample. One way to measure well-spaced-ness is discrepancy [Keller et al. 2012; Shirley 1991]. Another measure is *saturation*, which depends on two radii: coverage radius r_c for maximum domain to sample distance, and conflict radius r_f for minimum inter-sample distance [Ebeida et al. 2014; Mitchell et al. 2012]. Saturation is then quantified using their ratio $\beta = r_c/r_f$; the lower the β, the higher and better the saturation. Saturation is desired in many contexts, as described in the extensive literature on maximal Poisson-disk sampling (MPS) [Cline et al. 2009; Ebeida et al. 2011, 2012; Gamito and Maddock 2009; Jones 2006; Yan and Wonka 2013] and low discrepancy sampling [Ahmed et al. 2016]. For some applications, it is unclear how important saturation is as the dimension increases.

Despite the potential applications for high dimensional sampling, most sample-generation algorithms are low dimensional, in part because of the curse of dimensionality — many blue noise algorithms do not scale well to high dimensions (e.g. tiling [Ahmed et al. 2017, 2016; Kopf et al. 2006; Wang and Suda 2017]), especially when seeking high saturation. The sampling methods that scale well with dimension do not provide a guarantee of local saturation, while those providing local saturation have exponential complexity. The

algorithms closest to obtaining both of these goals are based on advancing-front [Bridson 2007; Liu 1991] or k-d darts [Ebeida et al. 2014], as detailed below.

2.1 Advancing front

Advancing front methods were initially proposed for meshing [Li et al. 2000; Liu 1991; Liu et al. 2008] and later adopted for sampling in graphics [Bridson 2007; Dunbar and Humphreys 2006; Jones 2006]. The basic idea is to draw new samples from regions around existing samples (the front) and expand towards the rest of the domain. Most methods build some form of the front boundaries explicitly, and some construct the union of spheres [Li et al. 2000; Liu et al. 2008]. These methods are intractable in high dimensions because the number of intermediate-dimensional faces grows factorially with dimension. In practice, implementing the geometric primitives for the constructions would be challenging as well.

In Point-Annulus [Bridson 2007], a key innovation is to represent the front boundary implicitly, by a list of sample disks touching the front. Point-Annulus does rejection sampling around a prior sample, selecting a point uniformly by volume from the $[r_f, r_c]$ annulus around it. The sample is removed from the front after a fixed number (30) of consecutive rejections. Its advantage is locality, mitigating the effects of domain size. This enables tractable runtime in high dimensions.[1] The single page sketch in Bridson [2007] does not analyze saturation by dimension. We postulate that the method guarantees that a large fraction of the annulus volume is saturated, but does not bound the uncovered volume outside all annuli. More significantly, we have discovered that its output has an undesirable artifact, a sharp discontinuity in the density of points at the outer boundary of annuli, as further demonstrated in Section 4.

2.2 Hyperplane sampling

k-d darts [Ebeida et al. 2014] uses hyperplanes for Poisson-disk sampling: 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. Its disadvantage is that it does not guarantee *local* saturation, because hyperplanes are selected *globally* from the entire domain. (Global dart throwing [Cook 1986; Dippé and Wold 1985] has similar issues.)

In principle, using hyperplanes of any dimension is possible, up to the dimension of the domain. However, the difficulty is actually performing and representing the necessary geometric primitives over this object. In k-d darts, only 1D lines and 2D planes were demonstrated. In the present work, we merely use lines, 1D hyperplanes. The method in Sun et al. [2013] samples lines and line-segments for rendering applications, including 3D motion blur, 4D lens blur, and 5D temporal light fields. For determining sample positions it relies on subroutines that do not scale well to high dimensions.

[1] As published, step 0 constructs a background grid. Replacing it with a k-d tree improves runtime from $2^{O(d)}O(n)$ to $O(dn^2)$, the same complexity as our spoke-dart sampling.

2.3 Combining advancing front with line search

Our key idea is to combine the advantages of advancing front and hyperplane sampling. Specifically, spoke-darts replaces the point-sampling of Bridson [2007] with line-sampling. A *spoke* is a line segment passing through a point, at a random orientation; see Figure 3. We employ a constant number (12) of consecutive rejections before advancing the front, retaining good run-time scalability across dimensions. However, 12 consecutive rejections provides a local saturation guarantee that is *the same in all dimensions*. We can generate different blue noise profiles. In particular, we can avoid the spike in the distribution at the sampling radius by non-uniform sampling from a spoke segment and by generating a second spoke through a point on the first spoke. These two spokes mimic the two radii in Mitchell et al. [2012], and produce a similar *soft blue noise* profile.

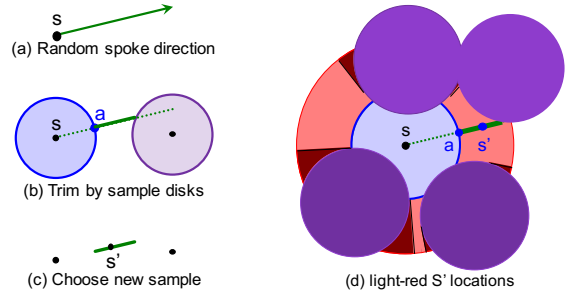


Fig. 3. Line-spokes in 2D. (a) A (green) spoke is a randomly-oriented line segment through a prior sample s. (b) It is trimmed by sample disks, keeping the (solid) subsegment containing anchor point a. (c) The next sample s' is chosen from the trimmed segment. (d) Because of the anchor point, the next sample will be in the subset of the annulus that is light-red, not the dark-red regions on the far side of other disks.

3 SPOKE-DART BLUE NOISE SAMPLING

Spoke-dart sampling generates new samples from the current sample set boundary and gradually expands towards the rest of the domain. The key features distinguishing our algorithm from prior methods are (1) how the front is described and advanced, and (2) how new samples are drawn. The front is described by the boundary of the union of disks around samples, but its geometry is not explicitly constructed. New samples are selected by generating a random spoke (radial line) through an existing sample, trimming it by existing sample disks, and selecting an uncovered point from the remaining sub-spoke.

Algorithm summary. Our top level algorithm follows. We initialize the output set with one sample and put it into the active pool of front points. When this pool becomes empty our algorithm terminates. We remove a 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 until $m = 12$ consecutive spokes failed to generate an acceptable sample. Radius r is the minimum allowed distance between any two samples. Our method is summarized in Algorithm 1.

Input: sample domain Ω
Output: output sample set $\mathcal{S}$

```
s ← RandomSample(Ω)
S ← {s} // all samples
P ← {s} // active pool, FIFO queue
while P not empty do
  s ← PopFront(P)
  N ← CollectNeighbors(s, S)
  reject ← 0
  while reject < m (=12) do
    ℓ, a ← RandomSpoke(s, I)
    ℓ ← TrimSpoke(ℓ, a, N)
    if ℓ is empty then
      reject ← reject + 1
    else
      s′ ← RandomSample(ℓ)
      if TwoSpokes then
        s′ ← SecondSpoke(s′, N)
      add s′ to N, S, and the end of P
      reject ← 0
return S
```

Algorithm 1. Spoke-darts for blue noise sampling.

3.1 Spokes

The line of a spoke passes through a sample s, and the spoke is the interval I of distances from s. A spoke has a distinguished *anchor* point $a \in I$ used to select which segment to retain during trimming; see Figure 3. For line-spokes, $I = [r, 2r]$ and the anchor lies at r, because, as in Point-Annulus, the uncovered region starts at r and the extent of the local front we wish to consider is $2r$.

3.2 CollectNeighbors

For a sample on the front, for each spoke we trim it by iterating over the nearby samples. For spoke-darts with spoke extent $2r$, a sample is a neighbor if its center distance is less than $3r$, because that is the farthest away a sample can lie and still have its disk overlap the spoke. A key efficiency is to gather all neighbors once *before* any trimming operations.

In our implementation, a k-d tree saves time over exhaustive search for small d and large n. Figures 9a and 18a show a speedup for $d < 7$ and $n \geq 200,0000$. We maintain a k-d tree of the entire point set. We collect the subtree of neighbors, and update the tree and subtree as we successfully add new samples.

3.3 RandomSpoke

A line-spoke is generated by selecting a line with random orientation, by choosing a point p from the surface of the disk around s, uniformly by area.

To pick p we use the classical method of Muller [1959], as follows. Generate each of the vector's d coordinates independently from a normal (Gaussian) distribution. Then linearly scale the vector of coordinates to the disk radius. The reason this works is because the level sets of a d-dimensional Gaussian distribution are d-spheres.

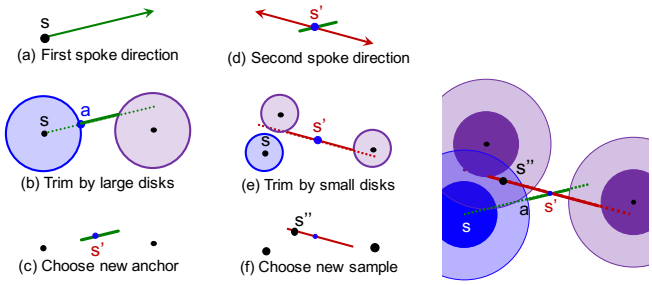


Fig. 4. (a)–(c) Two-spokes starts much the same as line-spokes, only using a longer spoke and trimming by $2r$-disks. (d)–(f) A second spoke is trimmed by r-disks, and the new sample s'' is chosen from it.

3.4 TrimSpoke

Trimming subtracts out the portion of a segment that is covered by a neighbor disk, leaving just its uncovered subset. For efficiency, we just keep the one subsegment that contains a distinguished *anchor point*; this is considerably faster than finding all uncovered segments. Further, we do a prepass and discard the entire spoke if the anchor is covered. These primitives are efficient, linear in dimension, and the prepass avoids square roots. In Figure 3, these shortcuts mean that the next sample will be chosen from the light red part of the annulus only, and not the dark red portions. This potentially affects the output distribution characteristics, but the saturation proof takes it into account.

3.5 RandomSample

For blue noise, it is sufficient to pick a sample *uniformly by length* from an uncovered spoke segment.

One might assume that picking a point uniformly by the swept volume, dependent on the dimension, would generate better quality blue noise. However, we found this detrimental for our algorithms, and also for the prior work of Point-Annulus [Bridson 2007]. It generates worse blue noise than traditional MPS algorithms; see Section 5.1. Exploring more sophisticated selection criteria led us to our two-spokes algorithm.

3.6 Two-spokes

Two-spokes is an algorithm variation that further randomizes the placement of samples; see Figure 4. Its output distribution avoids the traditional spike at the sampling radius, and mitigates other artifacts. We make the first spoke longer, and shoot a second spoke from an uncovered point on the first spoke. The first spoke has $I = [2r, 4r]$ with anchor at $2r$ and is trimmed by radius-$2r$ sample disks. The second spoke has $I = [-2r, 2r]$ with anchor at s' and is trimmed by radius-r sample disks. RandomSample is approximately uniform by swept volume from the nearer spoke end.

Two-spokes shares the following properties with two-radii Poisson-disk sampling [Mitchell et al. 2012]. Their spectra are similar, and the distance between samples is at least r. A new sample's large $2r$-disk covers s', and no other large disk covers it so far, ensuring progress and algorithm termination. There is a simple parameterization of the two spoke lengths that starts at line-spokes, then trades away saturation to gain randomness; see Appendix A.

4 ANALYSIS AND GUARANTEES

4.1 Probability of achieved saturation

Our measure of saturation is $\beta = r_c/r_f$, where r_c is the maximum distance from a domain point to its nearest sample, and r_f is the minimum guaranteed distance between a sample and its nearest sample, the Poisson-disk radius. Also, β^* is the desired upper limit on β. Besides spoke length, the main control parameter is m, the number of successively-failed spokes before removing a sample from the front. The higher the m, the more spokes we generate and the longer the run-time, but the more saturated the output. Note $(1-\epsilon)$ quantifies the probability that β^* is achieved. The structure of our guarantee is that, for a given m, with high probability $(1-\epsilon)$ the achieved β at a sample is less than β^*. Equation (1) quantifies the relationship between m, ϵ, β^*, and d for line-spokes.

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

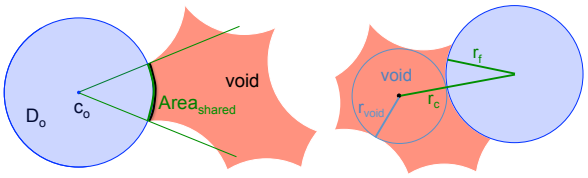
Our main result is that if $m = 12$, then with probability $1 - 10^{-5}$ we will get local $\beta < 2 = \beta^*$ in *any* dimension, avoiding the curse of dimensionality. In general, one can pick any three of $\{m, \epsilon, \beta^*, d\}$ and the fourth is determined. For example, one can pick m and β^* and bound the probability ϵ that β^* was exceeded: $\epsilon < \exp(-m(\beta^*-1)^{d-1})$, where $\beta^* > 1$, and $-\ln\epsilon > 0$, and $m \geq 1$.

- Line-spokes produces $\beta < 2$ with high probability.
- Two-spokes produces $\beta < 4$ with high probability.
  (The price of a more uniform spectrum is lower saturation.)

We provide some intuition for Equation (1) here; the derivations are in Section 4.2. Let us suppose that the algorithm has terminated and a sample has a far Voronoi vertex. Consider the empty ball centered at this vertex and tangent to the sample's disk; it lies outside all other samples' disks. We may expand this ball into a "void," a larger connected region bounded by sample disks. We have thrown at least m spokes from each of the void's bounding disks. Each of these spokes must have missed this void; otherwise we would have inserted a sample into the void, a contradiction. Since the void was not hit, if we add up its surface area across all bounding disks, its surface area is probabilistically-guaranteed to be small compared to the surface area of a single bounding disk. Thus the surface area of the Voronoi-vertex ball inside the void is also small, which bounds its radius. The exponential-in-$(d-1)$ dependence on β^* in Equation (1) is precisely the dependence of the surface area of a d-ball on its radius. Selecting $\beta^* = 2$ says we only care about voids with at least the surface area of a single sample disk. The exponential dependencies on surface areas cancel, and we are left with a Voronoi ball radius at most our sampling radius, meaning $\beta \leq 2$.

In practice, we achieve a much better saturation than the guarantee, $\beta \ll \beta^*$ for all m. This is expected because the proof is not tight: e.g., the void surface area might be much larger than that of an empty ball inside it, and we ignored chains of misses less than m.

While the bound on the probability of achieving $\beta < 2$ is dimension independent, the probability of achieving other β *does* depend on the dimension. Re-arranging Equation (1), we see $(\beta^*-1)^{d-1} = (-\ln\epsilon)/m$. For example, the probability of achieving $\beta < 1.5$ *decreases* rapidly with dimension, and the probability of achieving $\beta < 2.5$ *increases* with dimension. Thus, as d increases, we should expect the distribution of local betas to narrow and converge towards 2, or perhaps to some lesser constant due to the slack in the proof. This is what we observe in practice; see Figure 6 in Section 5.1, and also Figure 15 in Appendix B.



(a) shared area of a void and a disk (b) a ball with less surface area than its enclosing void

Fig. 5. Hitting a void from a neighboring disk.

4.2 Bound proofs

Here we prove the bounds on m, β, and ϵ in terms of d from Equation (1). 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. It will also depend on the surface area of any disk on its boundary, Area(D).

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 5a. 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 $p_1(\text{miss}) = 1 - R_1$. The chance of missing m times consecutively 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 $p_1^m(\text{miss}) < \exp(-mR_1)$.

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 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 $\exp(-mR) < \epsilon$, meaning $mR > -\ln(\epsilon) > 0$.

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 Area(void) > Area($D(r_{\text{void}})$); see Figure 5b. Since we are in d dimensions and $\beta = r_c/r_f$,

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

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

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

Two-spokes. If the first spoke finds an uncovered point, the second spoke always places a sample, so we need only consider the chance of the first spoke missing the void. The first spoke extends from $2r$ to $4r$. If we consider the subset of a void that is at least $2r$ from any disk, then propagating these values through the prior analysis shows $\beta < 4$ within that subset. The uncovered regions between r and $2r$ have local $\beta < 2$ and are subsumed, so the bound holds for the entire void. Hence $m = 12$ gives $\beta < 4$ with probability $1 - 10^{-5}$.

Subtleties. The reader may have noticed that we made no mention of the domain boundary. For bounded domains, we assumed that the void was bounded by disks only. For periodic domains, several analysis steps are only guaranteed to hold when the Voronoi-vertex ball spans less than the domain period. These issues may be finessed, e.g. by initializing with a few well-spaced samples.

There is another statistical subtly concerning the order of spokes. The consecutive misses from one bounding disk are not guaranteed to be consecutive with the misses from another disk. But this does not matter, because the misses for each disk 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 void that remains after the algorithm terminated.

5 EXPERIMENTAL RESULTS

5.1 Distribution comparisons

We compare the distributions of our methods and Point-Annulus experimentally. We provide open source software on the github repository SpokeDartsPublic [Awad et al. 2016], which may be used to verify the results. The original version of Point-Annulus [Bridson 2007] does not support periodic domains, so we re-implemented it as Point-Annulus*p for periodic domains in our framework.

Figure 7 shows the spectra, radial distance function (RDF), and radial power (RP) for Point-Annulus*p, line-spokes, and two-spokes, over 2–10 dimensional periodic domains. We conducted experiments in dimensions up to 30, but the figures for higher dimensions reveal no new structure or trends. Anisotropy is negligible because the algorithms do not depend on the choice of axes, e.g. all spoke directions are random, and sample neighborhoods are spheres. The only possible contribution to anisotropy is the fact that the domain's periodicity is axis aligned.

Many blue noise methods produce an RDF spike at r. However, for Point-Annulus, we were surprised to discover a discontinuity in the RDF at the outer annulus radius, $2r$, regardless of periodicity or implementation. For our methods, we notice a slight rise in RDF at $2r$ for line-spokes and some non-constantness even for two-spokes. These artifacts tend to decrease with dimension.

By design, the RDF and spectra of line- and two-spokes differ significantly. However, they have similar β distributions, after scaling by $\beta_{two} \approx 2\beta_{line}$, as described in Section 4. Figure 6 shows the median β by method and dimension.

- Line-spokes has median $\beta \approx 0.9$–1.2 as $d = 2$–5.
- Two-spokes has median $\beta \approx 1.8$–2.4 as $d = 2$–5.

The median β rises with dimension in part because of the increase in the number of Voronoi vertices around each sample, so the probability of at least one being far increases. However, recall from Section 4.1 that the distribution of achieved β narrows as the dimension increases, and should stay below a fixed value (< 2) as $d \to \infty$. Additional data presented in Figure 15 in Appendix B bear this out.

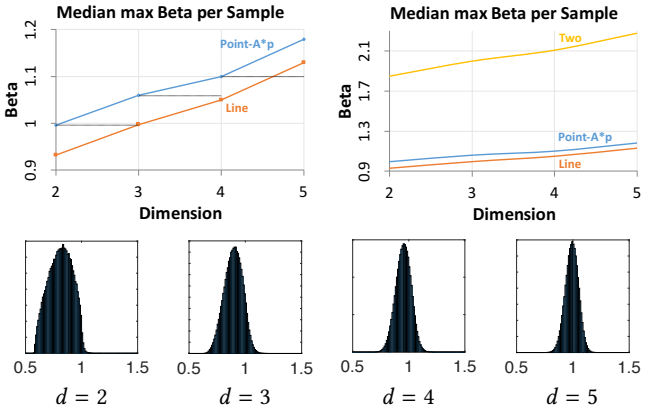


Fig. 6. Trends in β in practice. Top, line-spokes gives about the same saturation as Point-Annulus*p in one dimension lower. Two-spokes has about twice the β of Line-spokes by design. Bottom, the distribution of β (Voronoi-vertex to nearest-sample distances) narrows by dimension, and converges around a fixed value. We only show dimensions 2–5 because the available tools for computing Voronoi vertices, e.g. Qhull, run out of time and memory in higher dimensions.

5.2 Output size

We describe the relationship between the output number of samples n and the sampling radius r to allow the user to select the necessary r to achieve the desired n, for example. A sample point inhibits the introduction of nearby samples in a neighborhood related to distance r, so the volume of this neighborhood is roughly proportional to r^d.

$$n \approx k(d)/r^d$$

As d varies, the constant of proportionality k will vary, depending on the inherent packing density of the dimension [Weisstein 1999], and also because of our achieved β. Experimentally, for line-spokes,

$$k(d) = (0.46d + 1.8)1.04^d V_d,$$

where V_d is the volume of a unit d-sphere.

For two-spokes, the neighborhood around a point is roughly twice as large as line-spokes for the same r, so we expect n to be a factor of about $1/2^d$ smaller. In practice, $k_{\text{two}}(d) = (0.45d + 2.5)(1.04/2)^d V_d$. See Appendix B.1 for additional details.

5.3 Runtime scaling

We have three main observations:

- Runtime is linear in d for high dimensions, using exhaustive neighbor search. Albeit runtime is quadratic in n: $O(dn^2)$.
- Runtime is $\approx O(n \log n)$ for fixed d, using k-d trees.
- The crossover is about $d = 7$ for $n = 200,000$, meaning k-d trees are faster than exhaustive search for $d < 7$. The crossover dimension increases as n increases.

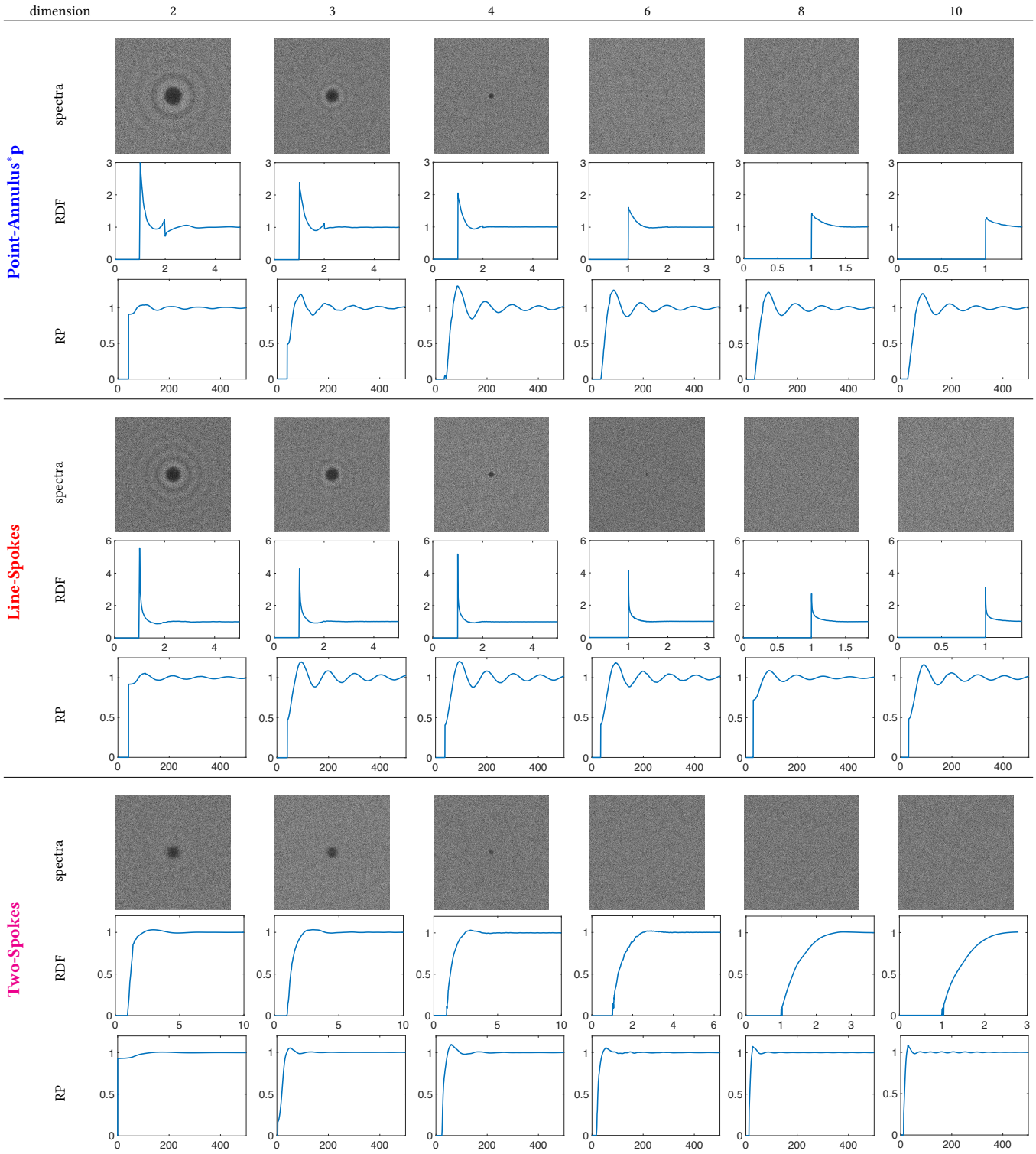
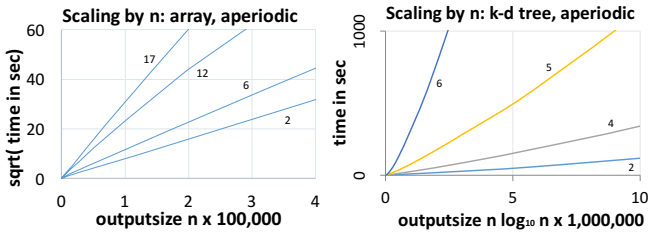


Fig. 7. Spectra, RDF, and RP for dimensions 2–10. Spectra for $d = 11$–30 are similar to $d = 10$. To keep the memory requirements tractable, spectral slices are computed directly in 2D using the Project-Slice Theorem (see, e.g. [Mersereau and Oppenheim 1974]). All plots use $n \approx 32{,}000$, except $d = 2$ uses more for smoother figures. RDF and RP were produced using the TargetRDF software [Heck et al. 2013]. For RP the DC component was filtered. Both RP and RDF were selectively smoothed and scaled. In RDF, "1" is scaled to r, the minimum distance between samples, and plots are truncated at absolute distance 0.5, to avoid the complication of the domain periodicity. For Point-Annulus and especially line-spokes, the RDF spike at r is sharp, and the plotted heights depend on the width and alignment of histogram bins, so exercise caution in drawing conclusions.

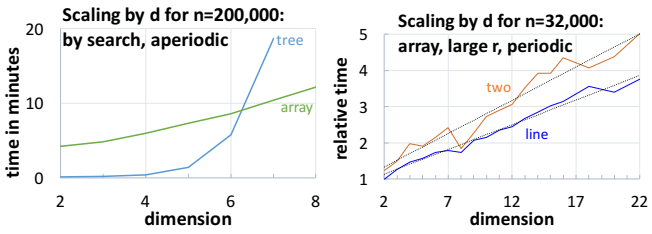
Complexity analysis. The runtime is $T = O(nF + dmNn)$, where n is the number of samples generated, and F is the time to find the N neighbors of a single sample. The $dmNn$ term represents the time to throw and trim spokes, including those that miss the void. Short sequences ($< m$) of miss spokes are charged to the next spoke that hits, just as in Bridson [2007]. For large d, we have $N \leq n$ and exhaustive search has $F = O(dn)$; thus $T = O(dn^2)$. For small fixed d, with $n \gg 2^d$ and $n \gg N$, using k-d trees $F \approx O(\log n + N)$ and $N = O(1)$; thus $T \approx O(n \log n)$.

Experiments. We verified these complexities experimentally. Figure 8 shows the predicted $O(n^2)$ and $O(n \log n)$ runtimes. Figure 9 demonstrates linear runtime in d using exhaustive search. Experimentally, the line-spokes runtime T using exhaustive search (array) over aperiodic domains is about $2.0 \times 10^{-9}(1 + 0.81d)n^2 + 5.5 \times 10^{-8}(1 + 0.05d)Nn + 2.4 \times 10^{-4}(1 + 0.05d)n$. Experimentally, the runtime for k-d tree search over periodic domains is about $7.8 \times 10^{-7} dn(0.12 \log_{10} n + N)$. See Appendix B.3 for additional analysis and experiments, including higher dimensions, periodic vs. aperiodic domains, and the number of neighbors by d.



(a) exhaustive array search (b) k-d tree neighbor search

Fig. 8. Line-spokes scaling by n for an aperiodic domain. Each trendline is labeled by the fixed dimension of the domain in that study. Left, straight trendlines illustrate $O(n^2)$ runtime for fixed d using exhaustive "array" search. Right, straight trendlines would illustrate perfect $O(n \log n)$ scaling for k-d trees.



(a) Runtime cross-over (b) Linear runtime in d

Fig. 9. Fixed-n scaling by d. Left shows that k-d trees save time in moderate dimensions. Right illustrates that runtime is linear in d for exhaustive array search. The right graphs are not smooth because we used only a few trials, and perhaps because of dimensional-dependent memory layout and machine issues, e.g. $d = 8$ appears particularly efficient.

6 APPLICATIONS

We demonstrate the versatility of the spoke-dart sampling approach, and the utility of its blue noise output. We briefly summarize each application below; additional details are in the Appendices. Delaunay graphs span $d = 6$–14, optimization $d = 6$–100, rendering $d = 4$–8, and motion planning $d = 6$–23.

The advancing-front spoke-dart sampling process provides new algorithms for approximate Delaunay graphs and global optimization. Our algorithm for approximate Delaunay graphs is significant because it avoids the curse of dimensionality and is dynamic. (By dynamic, we mean it can be updated quickly when inserting points, in contrast to some other known fast algorithms [Dwyer 1989].) We propose Opt-darts, a modification of the DIRECT global optimization algorithm [Jones et al. 1993; Shubert 1972]. Opt-darts uses the dynamic approximate Delaunay graph, and produces a well-spaced random output distribution of samples. For two standard test functions, we show that Opt-darts needs fewer function evaluations, and this speedup increases as the dimension increases.

Rendering and motion planning use our high-dimensional blue noise output directly as input. We show that high-dimensional rendering is possible, but using blue noise provides no apparent improvement over standard inputs. Being able to produce high-dimensional blue noise makes it feasible to run motion planning in high dimensions.

6.1 Approximate Delaunay graph

A Delaunay graph is just the edges in the Delaunay complex of a set of vertices (samples). These edges are dual to the $(d-1)$-dimensional facets of the Voronoi diagram. We find some of these facets, along with a point inside the facet. We shoot a spoke from a vertex, and trim it by each hyperplane separating the vertex from another vertex, retaining the hyperplane that trimmed it the most. The final spoke endpoint is a point inside a Voronoi facet, a witness to the fact that the facet exists in the Voronoi diagram. We tend to find the facets that subtend a large solid angle at the vertex, but miss some small facets. See Appendix C.1 for details.

6.2 Global optimization

The global-optimization algorithm DIRECT [Jones et al. 1993; Shubert 1972] is a classical and still-used method for optimizing expensive black-box functions, such as finite element simulation runs. It evaluates the objective function at each sample, and partitions the domain into hyperrectangles around each sample. A rectangle is recursively chosen for refinement if it is possible for the global minimum to lie inside it, assuming a fixed but unknown Lipschitz constant. Our variant, Opt-darts, partitions by Voronoi cells around each sample, instead of rectangles; these cells are implicit and only approximations are constructed. Opt-darts refines by adding new samples and updating nearby cell approximations. The new samples are chosen from among the spoke endpoints produced during the approximate Delaunay graph construction. Thus opt-darts uses both an approximate Delaunay graph, and generates an adaptive advancing-front random sampling.

In our tests, Opt-darts more accurately represents sample neighborhoods, and new samples are more well-spaced, so fewer of them

are needed. This advantage becomes more pronounced as the dimension increases. The disadvantage of Opt-darts is the higher computational cost in managing cells, but in the applications of interest the cost of the function evaluation at each sample dominates. In Table 1, we evaluate our method using community-standard high-dimensional test functions [Jamil and Yang 2013]. These were designed to be challenging for global optimization by having many local minima or a small gradient over most of the domain. Most difficult global optimization problems have some combination of these two features. For the Easom test function, in 6–10D speedups are 4–25×. For the Bohachevsky test function, in 20–100D speedups are 5–27×. See Appendix D for details.

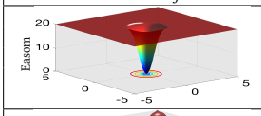
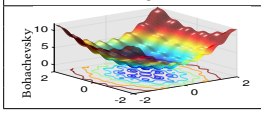
Benchmark f	dimension	DIRECT	Opt-darts	Speedup
	6	5657	1320	4.3 ×
	7	20987	3276	6.4 ×
	8	71677	4814	14.9 ×
	9	257539	14258	18.1 ×
	10	837203	33852	24.7 ×
	20	5689	1269	4.5 ×
	40	25807	2633	9.8 ×
	60	63765	4345	14.7 ×
	80	122503	6246	19.6 ×
	100	208185	7802	26.7 ×

Table 1. Speedup of Opt-darts over DIRECT, measured by the number of function evaluations needed to find an approximation f^* close to the true global minimum $\hat{f}$. That is: $|f^* - \hat{f}| < 10^{-4}$ where $f \in [0, 20]$. Since Opt-darts is random, results are averages over 20 runs.

6.3 Rendering

We integrated spoke-dart sampling into the Mitsuba renderer [Jakob 2010]. The torus-in-glass image (Figure 1c) demonstrates a bidirectional path tracer using 8D samples corresponding to 2D for the sky emitter, 2D for the camera screen, and 2D for each bounce along each camera and light path. Spoke-dart sampling, stratified sampling, and low-discrepancy sequences all produced images of similar quality. See Appendix E for details.

6.4 Motion planning

Motion planning explores the high-dimensional configuration space of robots to find collision-free paths between the given starting and desired ending configurations. In the "parallel RRT" algorithm [Park et al. 2016], this space is pre-sampled by blue noise, and multiple threads explore connect-the-dots paths. Sometimes, because the configuration space has narrow regions and fine features, the pre-sampling is too coarse to determine if the path between two nearby points is collision-free. In that case, fine blue-noise samples are adaptively added. We did motion planning for a challenging 23D problem (Figure 1d), and a well-known suite of 6D benchmark scenarios [Şucan et al. 2012]. See Appendix F for details.

7 CONCLUSIONS AND FUTURE WORK

We present spoke-dart sampling as a new framework for high dimensional sampling. The method combines the advantages of state-of-the-art methods: the locality of advancing-front and the dimension-mitigation of k-d darts, specifically line-samples. We provide spoke-dart sampling as open-source software [Awad et al. 2016]. To our knowledge, we provide the first algorithm for high dimensional blue noise with provable guarantees of local saturation. Line-spokes uses the same advancing front approach as Point-Annulus, but, by using line samples, it produces a median saturation about the same as Point-Annulus in one dimension lower. We also produce blue noise with less significant artifacts. We have the option to avoid the traditional distribution spike at the disk radius and corresponding oscillations in the spectra. We demonstrate spoke-dart sampling's generality by adapting it for a variety of applications, including generating high-dimensional adaptive blue noise for global optimization. Our algorithm uses linear memory, and is computationally efficient in high dimensions, up to the efficiency of finding nearby neighbors. We speculate that approximate nearest neighbors may improve scalability in moderate dimensions, but not high dimensions.

A potential future work is a universal algorithm that can automatically tune for a continuum of properties, analogous to Jiang et al. [2015]. It may be possible to produce a closer approximation to the true Delaunay graph by searching in a blue noise set of spoke directions. This could be generated by point-sampling the surface of a unit sphere, using spokes that are great-circle arcs. We speculate that approximate Delaunay graphs may be better than k-nearest neighbors for some computational topology and manifold learning problems, especially when data are non-uniformly spaced. High-dimensional global optimization is challenging, and Opt-darts demonstrates an improvement over DIRECT for two well-known test problems. Future research directions include cell selection criteria and parallelization. We briefly touched on using high-dimensional blue noise for rendering; there is the potential for future work in Monte Carlo integration [Pilleboue et al. 2015] and low discrepancy sequences [Keller et al. 2012]. In our current implementation for motion planning we precompute all samples. We are investigating the possibility of adaptive sampling by exploiting the similarity between our method and tree growth.

ACKNOWLEDGEMENTS

We thank the authors of Point-Annulus [Bridson 2007], TargetRDF [Heck et al. 2013], and PSA [Schlömer and Deussen 2011] for making their software available, and the reviewers for their helpful feedback and suggestions.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), Applied Mathematics Program. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

Ahmed, A. G. M., Niese, T., Huang, H., and Deussen, O. 2017. An adaptive point sampler on a regular lattice. *ACM Trans. Graph. 36,* 4 (July), 138:1–138:13.

Ahmed, A. G. M., Perrier, H., Coeurjolly, D., Ostromoukhov, V., Guo, J., Yan, D.-M., Huang, H., and Deussen, O. 2016. Low-discrepancy blue noise sampling. *ACM Trans. Graph. 35,* 6 (Nov.), 247:1–247:13.

Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., and Desbrun, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. Graph. 22,* 3 (July), 485–493.

Awad, M. A., Ebeida, M. S., Mitchell, S. A., Patney, A., Rushdi, A. A., and Swiler, L. P. 2016. SpokeDartsPublic open-source software. v. 1.0, https://github.com/

samitch/SpokeDartsPublic.

Balzer, M., Schlömer, T., and Deussen, O. 2009. Capacity-constrained point distributions: A variant of Lloyd's method. *ACM Trans. Graph. 28,* 3 (July), 86:1–86:8.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw. 22,* 4 (Dec.), 469–483.

Bossek, J. 2017. smoof: Single- and multi-objective optimization test functions. *The R Journal.*

Bridson, R. 2007. Fast Poisson disk sampling in arbitrary dimensions. In *SIGGRAPH '07: ACM SIGGRAPH 2007 Sketches & Applications.* 5.

Burkardt, J. 2011. TEST OPT: Optimization of a scalar function test problems.

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. *ACM Trans. Graph. 31,* 6 (Nov.), 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. *ACM Trans. Graph. 25,* 3 (July), 503–508.

Dwyer, R. A. 1989. Higher-dimensional Voronoi diagrams in linear expected time. In *Proceedings of the fifth annual Symposium on Computational Geometry.* SCG '89. ACM, New York, NY, USA, 326–333.

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,* 25–36.

Ebeida, M. S., Davidson, A. A., Patney, A., Knupp, P. M., Mitchell, S. A., and Owens, J. D. 2011. Efficient maximal Poisson-disk sampling. *ACM Trans. Graph. 30,* 4 (July), 49:1–49:12.

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., 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. *ACM Trans. Graph. 30,* 4 (July), 48:1–48:12.

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.

Heck, D., Schlömer, T., and Deussen, O. 2013. Blue noise sampling with controlled aliasing. *ACM Trans. Graph. 32,* 3 (July), 25:1–25:12.

Horst, R., Pardalos, P. M., and Romeijn, H. E. 2002. *Handbook of Global Optimization.* Vol. 2. Springer.

Jakob, W. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

Jamil, M. and Yang, X.-S. 2013. A literature survey of benchmark functions for global optimization problems. *Intl. Journal of Mathematical Modelling and Numerical Optimization 4,* 2, 150–194.

Jiang, M., Zhou, Y., Wang, R., Southern, R., and Zhang, J. J. 2015. Blue noise sampling using an SPH-based method. *ACM Trans. Graph. 34,* 6 (Oct.), 211:1–211:11.

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.

Jones, T. R. 2006. Efficient generation of Poisson-disk sampling patterns. *Journal of graphics tools 11,* 2, 27–36.

Kailkhura, B., Thiagarajan, J. J., Bremer, P.-T., and Varshney, P. K. 2016. Stair blue noise sampling. *ACM Trans. Graph. 35,* 6 (Nov.), 248:1–248:10.

Keller, A., Premoze, S., and Raab, M. 2012. Advanced (quasi) Monte Carlo methods for image synthesis. In *ACM SIGGRAPH 2012 Courses.* SIGGRAPH '12. 21:1–21:46.

Kollig, T. and Keller, A. 2003. Efficient illumination by high dynamic range images. In *EGRW '03.* 45–50.

Kopf, J., Cohen-Or, D., Deussen, O., and Lischinski, D. 2006. Recursive Wang tiles for real-time blue noise. *ACM Trans. Graph. 25,* 3 (July), 509–518.

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. 2005. A procedural object distribution function. *ACM Trans. Graph. 24,* 4, 1442–1461.

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.

Leong, K. Y. 2016. Test functions for optimization.

Li, H., Wei, L.-Y., Sander, P. V., and Fu, C.-W. 2010. Anisotropic blue noise sampling. *ACM Trans. Graph. 29,* 6 (Dec.), 167:1–167:12.

Li, X.-Y., Teng, S.-H., and Üngör, A. 2000. Biting: advancing front meets sphere packing. *International Journal for Numerical Methods in Engineering 49,* 1-2, 61–81.

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

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.

Lloyd, S. 1983. An optimization approach to relaxation labeling algorithms. *Image and Vision Computing 1,* 2.

Mersereau, R. M. and Oppenheim, A. V. 1974. Digital reconstruction of multidimensional signals from their projections. *Proceedings of the IEEE 62,* 10 (Oct), 1319–1338.

Miller, G. L. and Sheehy, D. R. 2013. A new approach to output-sensitive voronoi diagrams and delaunay triangulations. In *SoCG '13.* 281–288.

Miller, G. L., Sheehy, D. R., and Velingker, A. 2013. A fast algorithm for well-spaced points and approximate delaunay graphs. In *SoCG '13.* 289–298.

Mitchell, D. P. 1987. Generating antialiased images at low sampling densities. In *SIGGRAPH '87.* 65–72.

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.

Muja, M. and Lowe, D. G. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1).* 331–340.

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.

Niederreiter, H. 1992. *Random number generation and quasi-Monte Carlo methods.* SIAM.

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. *ACM Trans. Graph. 29,* 6 (Dec.), 168:1–168:8.

Öztireli, A. C. and Gross, M. 2012. Analysis and synthesis of point distributions based on pair correlation. *ACM Trans. Graph. 31,* 6 (Nov.), 170:1–170:10.

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. 2016. Parallel motion planning using Poisson-disk sampling. *IEEE Transactions on Robotics.*

Pilleboue, A., Singh, G., Coeurjolly, D., Kazhdan, M., and Ostromoukhov, V. 2015. Variance analysis for Monte Carlo integration. *ACM Trans. Graph. 34,* 4 (July), 124:1–124:14.

Reinert, B., Ritschel, T., Seidel, H.-P., and Georgiev, I. 2016. Projective blue-noise sampling. *Computer Graphics Forum 35,* 1, 285–295.

Schechter, H. and Bridson, R. 2012. Ghost SPH for animating water. *ACM Trans. Graph. 31,* 4, 61:1–61:8.

Schlömer, T. and Deussen, O. 2011. Accurate spectral analysis of two-dimensional point sets. *Journal of Graphics, GPU, and Game Tools 15,* 3, 152–160.

Shirley, P. 1991. Discrepancy as a quality measure for sample distributions. In *Eurographics '91.* 183–194.

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

Subr, K. and Kautz, J. 2013. Fourier analysis of stochastic sampling strategies for assessing bias and variance in integration. *ACM Trans. Graph. 32,* 4 (July), 128:1–128:12.

Sun, X., Zhou, K., Guo, J., Xie, G., Pan, J., Wang, W., and Guo, B. 2013. Line segment sampling with blue-noise properties. *ACM Trans. Graph. 32,* 4 (July), 127:1–127:14.

Ulichney, R. A. 1988. Dithering with blue noise. *Proceedings of the IEEE 76,* 1, 56–79.

Wang, T. and Suda, R. 2017. Fast maximal poisson-disk sampling by randomized tiling. In *HPG '17.* 16:1–16:10.

Wei, L.-Y. and Wang, R. 2011. Differential domain analysis for non-uniform sampling. *ACM Trans. Graph. 30,* 4 (July), 50:1–50:10.

Weisstein, E. W. 1999. Hypersphere packing. http://mathworld.wolfram.com/HyperspherePacking.html.

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 Trans. Graph. 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.

Yang, X.-S. 2010. *Appendix A: Test Problems in Optimization.* John Wiley and Sons, Inc., 261–266.

A SOFT BLUE NOISE

We seek a better blue noise spectrum than line-spokes or Point-Annulus produces, where "better" is defined in the following sense. The main drawbacks of those distributions is a large spike in the inter-sample distances near r, and corresponding oscillations in the radial power. To remove this, we first consider changing the distribution for choosing a random sample from a spoke to be more uniform by volume. This, by itself, proved insufficient to remove the spike. We experimented with additional rules such as skipping short spokes; see favored-spokes Appendix A.1. Although this algorithm has a unique advantage of a median saturation that is invariant by dimension, it should mostly be considered a stepping-stone towards two-spokes in Appendix A.2. We found that taking a second spoke was simpler and more intuitive than the skip rules, and generally produced a distribution with a flatter spectrum.

A.1 Favored-Spokes

To generate step blue noise, we use the same top level algorithm: Algorithm 1. However, we seek to *avoid the spike* in the RDF distribution at r. We use different rules to accept and sample from spokes:

- place samples on short trimmed spokes less often,
- place samples nearer the center of uncovered intervals.

This reduces the number of disks nearly-r apart, but does not eliminate them because a spoke might be close to a disk without intersecting it. We must avoid any sharp cut-off values in the rules, because these would create new discontinuities in the RDF. We arrived at the following ranges experimentally.

Skip short spokes. We use spoke interval $I = [1, 3.8]r$. We never place a sample point farther than $3.4r$, but the spoke extends to $3.8r$ so we can detect if a sample point would be near an extant disk.

We have two skip rules. The first rule is if a spoke is trimmed by any disk, then we discard it and treat it as a miss. We do this until we have 6 successive misses. After this we reset the miss count to zero and apply the second rule until we again get 6 successive misses. See the *open* and *closed* sectors in Figure 10.

The second rule is spokes are discarded if they are *short*. Spokes are discarded if their extent is less than $3.2r$ (length $< 2.2r$), and randomly discarded with decreasing probability if their extent is in $[3.2, 3.5]r$; i.e. always discarded if within the dark blue ring, and sometimes discarded if in the light ring, in Figure 10. The discard probability is zero at 3.5 and grows cubically to 1 at 3.2. Experimentally, a cubic rate produced better output than a linear or dimensional-dependent rate. Algorithm 2 describes TrimSpoke with these rules in place.

Randomize spoke endpoints. We shorten the ends of a spoke by a random amount in $[0.3, 0.8]r$ to avoid sharp cutoff values.

RandomSample for favored-spokes and two-spokes. A spoke actually trimmed by a neighbor is *closed*; one with no disk intersections is *open*. For closed intervals, we place the sample point approximately uniformly by volume by the distance to the *nearer spoke end*. This underweights the volume near the front sphere. For open segments, we place the sample approximately uniform by volume by distance to the end near the anchor, then flatten off and ramp down; see Figure 11.

Input: line spoke ℓ_1 anchored at a for sample s
Output: uncovered segment of trimmed spoke ℓ_1'

```
if TrimAnchor(a) = empty then
    return empty
ℓ1' ← TrimInterval(ℓ1) // not empty
if SkipTrimmedSpoke and WasTrimmed(ℓ1') then
    if reject = 6 then
        SkipTrimmedSpoke ← false
        reject ← 0
    return empty
if IsShort(ℓ1') then
    return empty
return ℓ1'
```

Algorithm 2. TrimFavoredSpoke, for step blue noise.

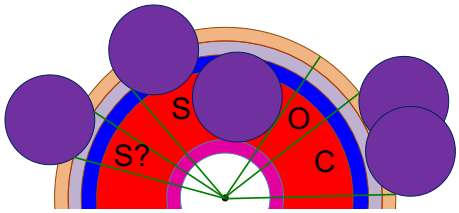
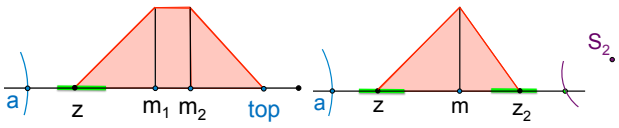


Fig. 10. Favored-spokes ranges. Spokes in the "O" sector are open and "C" are closed; "S" are short, and "S?" are considered short with some probability. The following are the outer ring radii as a factor of r: white = 1, magenta = 1.3, red = 2.8, blue = 3.2, light blue = 3.5, brown = 3.8.



(a) Open spokes (b) Closed spokes

Fig. 11. Non-uniform sampling for favored-spokes and two-spokes. The sample is chosen uniformly by volume under the red curves. In 2d, from z the curve is linear with slope 1 until m. In general, $y = (x - z)^{d-1}$. (b) For closed spokes, the midpoint m is 0.6 of the way from z to z_2. (a) For open spokes, $m_1 = 0.54$ and $m_2 = 0.7$. For two-spokes the z and top are the spoke endpoints. For favored-spokes the z are chosen uniformly to be distance $[0.3, 0.8]r$ from the end of the spoke, on the green segments, and the top at distance $3.4r$ rather than $3.8r$.

A.2 Two-Spokes for soft blue noise

Two-spokes also uses the same top level algorithm: Algorithm 1. We generate an uncovered point as with line-spokes, without the skip rules of favored-spokes. We pick the final sample by taking a second spoke through this uncovered point; see Figure 4. This mimics our prior Two Radii [Mitchell et al. 2012] sampling: the first spoke mimics finding a point uncovered by prior $2r$ disks, and the second spoke mimics the larger admissible region for centers of disks that can cover it.

First spoke. The first spoke extends from $2r$ to $4r$, with anchor at $2r$, and is trimmed by $2r$ disks. We select the uncovered point s' from it using the RandomSample of favored-spokes.

Second spoke. The second spoke is centered on, and anchored by, s' from the first spoke, with $I = [-2, 2]r$. After it is trimmed by $1r$ disks, we split it by s' into two sides. We pick one side uniformly by length, and select the final sample from that side's segment using RandomSample from favored-spokes. See Algorithm 3.

Input: uncovered point s'
Output: output nearby uncovered point s''
1: $\ell \leftarrow$ RandomSpoke($s', I = [-2, 2]r$)
2: $\ell \leftarrow$ TrimSpoke($\ell, a = s', N(s')$) // never empty
3: **return** $s'' \leftarrow$ RandomSample(ℓ)

Algorithm 3. SecondSpoke, generating a random uncovered point near the input uncovered point.

Generalization. Two-spokes may be generalized to give the option to soften the step in the RDF distribution by different amounts, by parameterizing by $\alpha \in [0, \text{inf})$ and $\gamma \in [0, 1]$. Line-spokes corresponds to $\alpha = 0$ and $\gamma = 0$, and two-spokes as previously described corresponds to $\alpha = 1$ and $\gamma = 1$. The first spoke has $I = (1+\alpha)r[1, 2]$ and $a = (1 + \alpha)r$ and is trimmed by radius $(1 + \alpha)r$ sample disks. The second spoke has $I = \gamma(1 + \alpha)r[-1, 1]$ and $a = 0$ and is trimmed by radius r sample disks. This will ensure that the first spoke finds an uncovered point, at least $(1 + \alpha)r$ away from all prior samples, and the chosen sample will cover it by its $(1 + \alpha)r$ disk. The chosen sample will be at least $r_f \geq r$ away from any other sample. For small γ, the intersample distance will be even larger, by the triangle inequality: $r_f \geq (1 - \gamma)(1 + \alpha)$. Thus $r_f \geq \max(r, (1 - \gamma)(1 + \alpha))$. The form of the β^* guarantee is that r_c is likely at most twice the anchor distance of the first spoke: $r_c \leq 2(1 + \alpha)r$. Thus for $m = 12$, with probability $1 - 10^{-5}$ we have

$$\beta < \beta^* = 2 \min\left((1 + \alpha), \frac{1}{1 - \gamma}\right) \tag{2}$$

B ADDITIONAL EXPERIMENTAL RESULTS

In this appendix, we show further comparisons between Point-Annulus, line-spokes, favored-spokes, and two-spokes.

B.1 Output size data

Section 5.2 gave approximate formulas for the number of output samples n by radius r and dimension d for line-spokes and two-spokes for periodic domains. Recall $n \approx k(d)/r^d$ with a different $k(d)$ for each algorithm. For line-spokes over periodic domains of dimensions 2–22, we have $k_{\text{l,p}}(d) = (0.46d + 1.8)1.04^d V_d$, where V_d is the volume of a unit d-sphere. For two-spokes we have $k_{\text{two}}(d) = (0.45d + 2.5)(1.04/2)^d V_d$.

For bounded domains, part of a sample's neighborhood falls outside the domain, so we expect the same r to produce a larger n. In practice, for line-spokes over dimensions 2–30, we observe $k_{\text{l,b}}(d) \approx (0.0004d^4 - 0.027d^3 + 0.52d^2 - 2.5d + 6.2)V_d$.

The output size trends for favored-spokes are similar to line-spokes, but with the constant of proportionality $k(d)$ having less dimensional dependence. Experimentally, we observe

$$k_{\text{fav}}(d) = (0.035d + 1.15)1.04^d V_d,$$

See Figure 12 for a summary. In addition, Figure 18d shows how the radius varies across dimensions for fixed n for our algorithm variants.

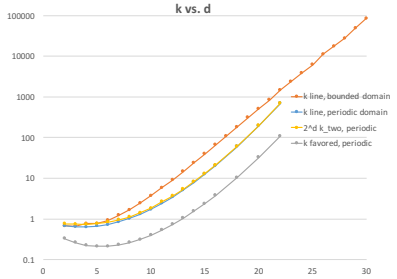


Fig. 12. Output size constants of proportionality. Number of samples $n \approx k(d)/r^d$ where $k(d)$ is plotted vs. d.

B.2 Distribution

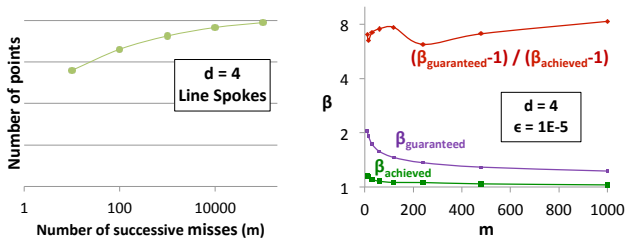
Figure 13 shows how increasing m affects the output saturation. As the dimension increases the distributions tend to resemble a Gaussian and become more sharply peaked. While the theory guarantees are invariant in dimension, in practice line-spokes and two-spokes produce slightly larger β in higher dimensions.

Figure 14 shows how the median beta varies by dimension, for $m = 12$ and all methods. Figure 15 provides distribution details beyond the median. Specifically, it shows the distribution of the distance from each sample to its farthest Voronoi vertex and the distribution of distances from each Voronoi vertex to its nearest samples, using our algorithm's variations as well as Bridson's for $d = 2, 3, 4, 5$.

For aperiodic domains the boundary significantly affects the distribution characteristics, especially for coarse samplings, so these results are mostly omitted. The exception is Figure 16, which shows the spectra for Bridson's implementation run over aperiodic domains.

Besides supporting periodic domains and k-d tree search, the implementations differ in the order in which the front is advanced. In our implementation samples become the active front disk in the order in which they were created. We continue to sample around the active disk until 30 consecutive misses, then remove the disk from the front. In contrast, Point-Annulus visits front disks in random order: remove a random point from the queue as the active front disk, throw exactly 30 darts, then reinsert the front disk into the queue if any new sample was accepted. In our comparisons, this algorithmic difference does not affect the output distribution significantly.

Although we can create blue noise distributions in higher dimensions, our ability to analyze their β is limited by the challenge of computing Voronoi vertices. We use QHull, and for large d and reasonable n, QHull runs out of memory (and time).



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

Fig. 13. Local saturation for line-spokes in theory and practice by m. Here $\beta_{guaranteed} = \beta^*$, the probabilistically-guaranteed saturation upper-bound in theory. And $\beta_{achieved}$ is the β observed in experiments. In practice β is about 7× closer to 1 than the theory guarantee: e.g. $\beta \approx 1.15$ for $m = 12$.

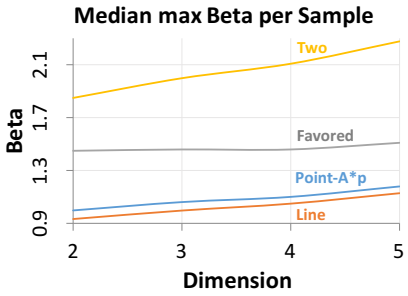


Fig. 14. Trends in median β for $m = 12$ by d.

B.3 Runtime scaling details

Complexity analysis details. The runtime complexity is $O(nF + dmNn)$, where n is the number of samples generated, and F is the time to find N neighbors. All primitives such as computing distances and trimming spokes are linear in d. Note N has dimensional dependence, but this is bounded by observing $N < n$. Also F has additional dimensional dependence if one uses k-d trees or other techniques, but this is bounded by brute force searching $F = O(dn)$. Hence runtime is $O(dmn^2)$. The $O(dmNn)$ term arises from generating spokes and trimming. For each sample we have one chain of m consecutive "miss" spokes that do not generate a new sample, plus some shorter miss chains that end with a spoke that generates a sample. To bound the overall run-time, we must account for these small chains. We assign the cost of a small chain to the sample disk insertion following it, not the front 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 $2mn$ spokes, each of which takes $O(dN)$ to trim.

Experiments set up. We verified this complexity experimentally. Calculations were performed on a mid-2010 Mac, with 3.33 GHz 6-Core Intel Xeon processor, and 16 GB RAM.

Experiments by output size. Figure 17 shows the scaling of line-spokes by n. Experimentally, the line-spokes runtime using exhaustive search (array) over aperiodic domains is about $2.0 \times 10^{-9}(1 + 0.81d)n^2 + 5.5 \times 10^{-8}(1 + 0.05d)Nn + 2.4 \times 10^{-4}(1 + 0.05d)n$. Experimentally, the runtime for k-d tree search over periodic domains is about $7.8 \times 10^{-7} dn(0.12 \log_{10} n + N)$.

Experiments across search type. Figure 18a compares the runtime of all the line-spoke variations in small dimensions. Dimension 3 is close to as fast as dimension 2, because in all cases neighbor searching is a small fraction of the total time. The advantages of a tree vs. exhaustive search disappear at dimension 6 or 7 for $n = 200,000$. For smaller n, the advantages disappear sooner.

Neighbors and periodicity. Figure 18b shows the exponential increase in the number of neighbors by dimension for periodic domains. It also shows the effect this has on the runtime of the favored- and two-spoke k-d tree variants, mainly due to their longer spokes. For aperiodic domains, the boundary strongly effects N as n varies, and is not illuminating.

Neighbor searching with exhaustive search is not much more expensive for periodic domains than aperiodic domains, but for k-d tree search periodic domains are increasingly expensive as the dimension increases. For k-d tree search, in the worst case one must do a tree search for each periodic translation of the query point, multiplying the query time by 2^d. This happens more frequently for small n (large r). However, with larger n, there is additional expense in rebalancing the tree.

For exhaustive search and small radius, the only extra step is to find the periodic translation of each coordinate of the test point that is closest to the query point, only adding a small $O(d)$ term. For large radius, a spoke may cross more than one periodic copy of a sphere. To trim a spoke, we march numerically along the spoke from the anchor, ensuring that we are trimming by the closest periodic copy of each sphere at each step. This increases the runtime by a factor of about 50; this is large, but still a constant across d and n.

Experiments by dimension. Figure 18c shows linear scaling in d for fixed n using exhaustive search.

C APPLICATION: APPROXIMATE DELAUNAY GRAPHS

C.1 Motivation

Many applications rely on knowing the nearby neighbors of points. Often it is not enough to know just the nearest point or those within some threshold. A Delaunay graph describes both nearness and directionality; intuitively it provides all points that are nearest in some general direction. The size of the full Delaunay tessellation (including faces of all dimension) is inherently exponential in dimension, and computing it becomes intractable [Barber et al. 1996]. However, the number of edges is at most $O(n^2)$. For some types of random input, the number of edges is expected to be linear, $E(n)$, and these can be found in $E(n)$ time for static input [Dwyer 1989]. For meshing and other algorithms, the input is not static and as points are adaptively added the Delaunay graph must be dynamically updated. Some recent theoretical papers [Miller and Sheehy 2013; Miller et al. 2013] have considered dynamic approximate graphs and the challenges of high dimensions from the standpoint of complexity analysis, although no implementations or experimental results for these algorithms are available, and they require an over-approximation rather than an under-approximation as we provide.

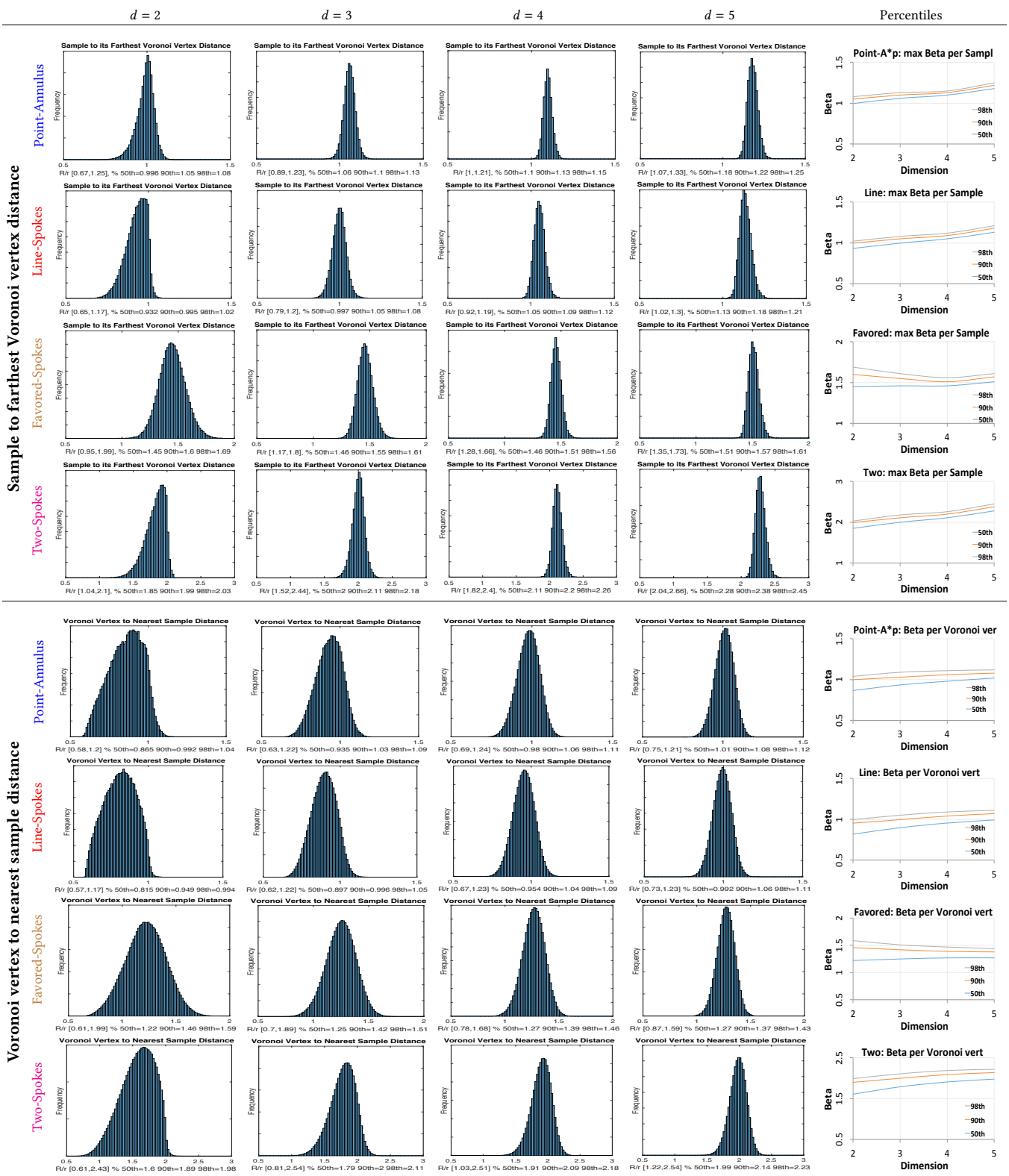


Fig. 15. β distribution histograms: (top) sample to farthest Voronoi vertex distance, and (bottom) Voronoi vertex to nearest sample distance, for d =2–5. The bottom-half rightmost-column illustrates that, in practices, β narrows and converges to a fixed constant as d increases, as theory predicts.

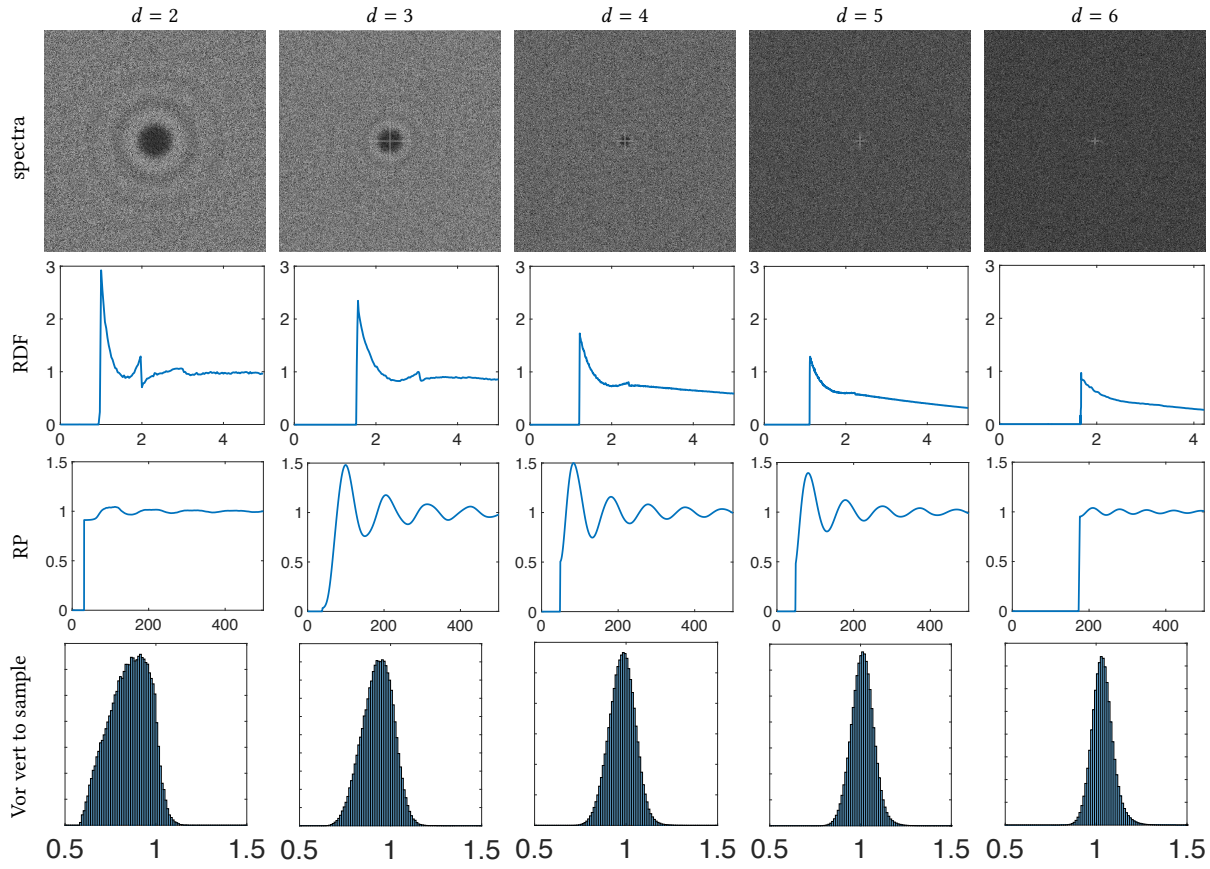


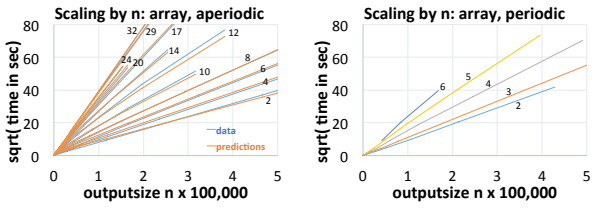
Fig. 16. Point-Annulus output across dimensions 2–6 for aperiodic domains.

To our knowledge, we present the first practical implementation for dynamic approximate Delaunay graphs in high dimensions. Graph $\mathcal{D}^*$ 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*. As a further benefit, our method produces a *witness* for each edge, a domain point on its true Voronoi face, which can be used to estimate the radial extent of the Voronoi cell. This is used in our global optimization application, Appendix D.

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 can accept approximate Delaunay graphs; the effect of the missing edges is application dependent. For global optimization, Appendix D, the approximation affects efficiency and not correctness. The neighborhood sizes around sample points determine the order in which new samples are generated. Prior approaches use rectangles which usually grossly overestimate the neighborhood sizes. The significant Delaunay graph provides a more accurate estimate of neighborhood sizes. The true Voronoi vertices of a true Delaunay graph would give the most accurate sizes, but there are an exponential number of them. Such compromises are commonly necessary for high-dimensional problems. For example, the state-of-the-art high-dimensional nearest neighbor (and k-nearest) query [Muja and Lowe 2009] often returns the wrong nearest point, but its distance is probably not much greater than the distance to the true nearest neighbor. As algorithms for high-dimensional graphs have improved, their use has increased in fields such as uncertainty quantification [Witteveen and Iaccarino 2012] and computational topology [Gerber et al. 2010].

C.2 Algorithm

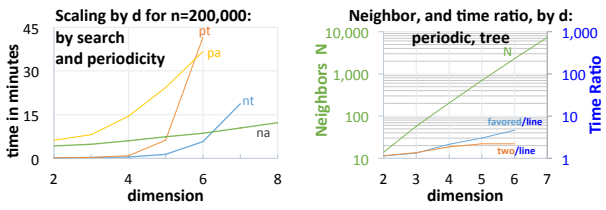
Our basic idea is to throw random line-spokes to tease out the significant Delaunay edges from a set of spatial neighbors. This is a very simple method that scales well across different dimensions. It is summarized in Algorithm 4 with details as follows. We construct the graph $\mathcal{D}^*$ for each 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 degenerate cases where multiple vertices trim the spoke the most and equally, we can pick an arbitrary one for s^*.) The far end of the trimmed spoke ω is equidistant from s and s^*, and no



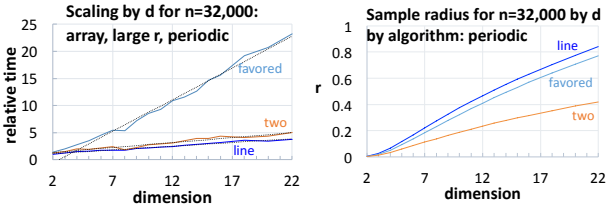
(a) aperiodic domain (b) periodic domain

(c) aperiodic domain (d) periodic domain

Fig. 17. Line-spokes scaling by output sample number n. Top: exhaustive array neighbor search. Bottom: k-d tree search. Each trendline is labeled by the dimension of the domain in that study. In the top, the trendlines being straight illustrate that using exhaustive search runtime is $O(n^2)$ for large but fixed d. In the bottom, straight trendlines would illustrate perfect $O(n \log n)$ scaling.



(a) Runtime cross-over (b) Neighbor scaling



(c) Linear runtime in d (d) Radius for fixed n by d

Fig. 18. Fixed-n scaling by dimension. In (a), "pt" is periodic domain with tree search, and "na" is non-periodic with exhaustive array search, etc. The straight lines in 18c demonstrate linear runtime in d for array search.

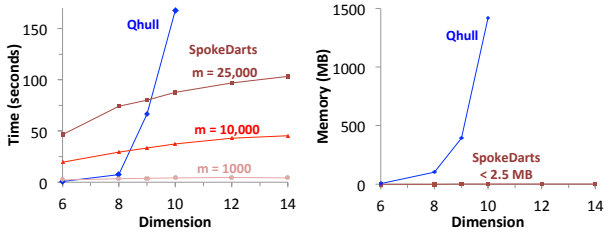
other vertex is closer. Hence ω is the witness that s and s^* share a Voronoi-face (Delaunay-edge), and $\overline{ss^*}$ is added to $\mathcal{D}^*$.

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 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 $\mathcal{D}^* \to \mathcal{D}$.

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

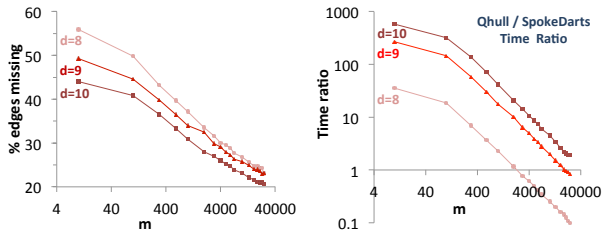
1: // R = true for a new vertex s
2: $\mathcal{N} = \emptyset$ // approx. Delaunay neighbors of s
3: $\delta(s) = 0$ // approx. cell radius of s
4: **for** $i = 1$ **to** m **do**
5: $\quad \ell \leftarrow \text{RandomSpoke}(s, 0, |\Omega|)$
6: $\quad$ **for** each sample $s' \in \mathcal{M}$ **do**
7: $\quad\quad \pi(s, s') \leftarrow$ hyperplane between s and s'
8: $\quad\quad$ trim ℓ with $\pi(s, s')$
9: $\quad\quad$ **if** ℓ got shorter **then**
10: $\quad\quad\quad s^* \leftarrow s'$
11: $\quad \mathcal{D}^* \leftarrow \mathcal{D}^* \bigcup \left\{\overline{ss^*}\right\}$ // without duplication
12: $\quad \mathcal{N} \leftarrow \mathcal{N} \bigcup \{s^*\}$
13: $\quad \delta(s) = \max(\delta(s), \text{length}(\ell))$
14: **if** R = true **then**
15: $\quad$ // update edges of neighbors, removing some
16: $\quad$ **for** each sample $s' \in \mathcal{N}$ **do**
17: $\quad\quad \mathcal{M} \leftarrow \text{Neighbors}(s') \bigcup \{s\}$
18: $\quad\quad \mathcal{D}^* \leftarrow \mathcal{D}^* \setminus \text{Edges}(s')$ // remove all edges
19: $\quad\quad \text{Recurse}(s', \mathcal{D}^*, \mathcal{M}, \text{false})$ // restore some
20: **return** $\mathcal{D}^*$

Algorithm 4. Add a vertex to the approximate Delaunay graph.



(a) computation time (b) memory requirement

Fig. 19. 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

Fig. 20. Effects of m on the approximate Delaunay graph. As m increases, fewer Delaunay edges are missed (a) but run-time increases (b).

C.3 Experiments

We demonstrate the efficiency of our approach against Qhull [Barber et al. 1996]. It is a commonly-used code for *full* convex hulls and

Delaunay triangulations, and hence suffers from the curse of dimensionality. We know of no tools for approximate Delaunay graphs to compare against. 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 19 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 20 shows the effect of m on the time and number of missed edges.

D RETHINKING GLOBAL OPTIMIZATION USING VORONOI DECOMPOSITIONS

A variety of disciplines, — science, engineering or even economics, — seek the "best" answer for a question under study. This usually requires solving a global optimization problem, where we have to explore the parameter space of a function f in order to find the optimum point $\hat{f}$ of some objective function, possibly under a set of feasibility constraints. For many problems, local optimality is not sufficient and a global optimal point is desired. 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. In practice, heuristic stochastic techniques are usually the best, and sometimes the only option [Horst et al. 2002]. Lipschitzian optimization is an important category of global optimization methods. Shubert [1972] explores the parameter space and provides convergence based on the Lipschitz constant K of the objective function, where a function f is *Lipschitz continuous* with constant $K > 0$ if

$$|f(x_i) - f(x_j)| \leq K|x_i - x_j| \tag{3}$$

for all $x_i \neq x_j$ in the domain of f. The DIRECT algorithm [Jones et al. 1993] extends Shubert's work to higher dimensions and does not require knowledge of K, decomposing a domain into nested hyperrectangle partitions.

In this section, we demonstrate how spoke-dart sampling can further generalize DIRECT by sampling points in random directions, not necessarily aligned with grid lines and replacing the nested hyperrectangle partitions with *random approximations* to Voronoi cells. Using classical test functions, we briefly illustrate how spoke-dart sampling has a significantly improved optimization performance. We believe this opens the door to new solutions of optimization problems. Below, we outline our optimization approach, called "Opt-darts" and provide a careful comparison between Opt-darts and DIRECT.

To our knowledge, our method is the first stochastic Lipschitzian optimization technique. Our use of the phrase "stochastic" refers to the randomness in the Voronoi cell seed locations within our algorithm; we are not referring to optimization of a stochastic objective function. Computing and refining random Voronoi cells has been intractable in high dimensions due to the exponential growth of Voronoi vertices as the dimension increases, and this is probably why this direction has not been explored before. Our spoke darts algorithm enables the size estimation of the Voronoi Cells without explicitly calculating and storing these Voronoi vertices. This allows tractable cell refinement needed in solving global optimization problems.

D.1 DIRECT algorithm

The DIRECT (DIviding RECTangles) algorithm [Jones et al. 1993] was developed for optimization of "black-box" functions (often expensive engineering simulations) which may be nonlinear, nonconvex, and multi-modal. DIRECT is a global optimization approach that combines global exploration of the space with local search around the best solution and does not require gradient information. DIRECT partitions the domain into hyperrectangles. It refines those rectangles, typically by trisecting each rectangle along one of its long sides. The refinement process creates nested hyperrectangles that could contain a point whose function value is smaller than the smallest f^* found so far. This refinement recurses until reaching the maximum number of iterations, or the remaining possible improvement is small. An important aspect of DIRECT is that it does not just pick one hyperrectangle for refinement. Instead, several hyperrectangles are selected based on relative weightings of local versus global search.

Each rectangle i is associated with two quantities: 1) a function evaluation f_i at its center and 2) a size estimate h_i given by the distance from the rectangle center c_i to any of its corners. The lower convex hull $\mathcal{H}$ of the 2D data points $\{h_i, f_i\}$ lists the cells to be refined next. This convex hull is a Pareto curve that represents the tradeoff between local search (search around the best values of f_i) and global search (search around points with large h_i because they have not been refined much yet).

To avoid overrefining the cell with the current best solution f^*, an artificial data point is added with $\{h_0 = 0, f_0 = f^* - \epsilon|f^*|\}$, where ϵ (typically set to 10^{-4}) is a parameter to balance global and local searches. A cell is refined by choosing an axis-aligned direction, and splitting the cell into three equal-sized cells in that direction. $\mathcal{H}$ is updated every time its cells are refined. This refinement recurses until the sample budget is exhausted. Note that limiting the cell refinement to those in $\mathcal{H}$ explores the most probable locations for a new best solution without any assumptions about the Lipschitz constant K of the underlying function.

D.2 Opt-darts algorithm: our method

In this section, we first highlight limitations of DIRECT and how they are addressed in Opt-Darts. We then present the details of how Opt-darts chooses the first sample, estimates a cell size, and refines a cell. We then summarize the algorithm in a pseudocode.

Motivation. DIRECT had a number of algorithmic limitations: 1) a cell can only be a hyperrectangle, 2) nested refinement, where a new cell can not extend beyond the boundaries of the refined cell, and 3) new sample points can only be on an axis-aligned direction with the refined cell's center. To mitigate these limitations, our algorithm (Opt-darts) uses Voronoi cells rather than hyperrectangles. From a Lipschitzian perspective, the cell size h_i should be the distance from its sample point c_i (cell seed) and its furthest Voronoi vertex v_i. This

offers a much more accurate neighborhood representation. On the other hand, nested refinement may result in false convergence (a phenomenon often reported by DIRECT users). This happens when DIRECT persistently refines a cell that is close to $\hat{f}$ but does not actually contain it; see Figure 21 for an illustration. Voronoi cells do not follow the nested refinement approach; each new sample includes the domain points closest to it, with no boundary constraints. In DIRECT, axis-aligned sampling results in a stair-pattern marching towards the global solution. Alleviating this constraint increases the possibility of sampling points closer to $\hat{f}$. Moreover, while DIRECT adds new samples in the interior of the refined cell, Opt-dart has the flexibility of adding points on the refined cell's boundaries. This is more efficient for space filling.

Algorithm. The Opt-darts algorithm is summarized in Algorithm 5.

Sampling first point. We pick the first point uniformly randomly from the middle 1/3 of the domain.

Cell size estimation. Starting at a cell seed c_i, we throw two sets of spokes: d spokes in axis-aligned directions (mimicking DIRECT), and $2d$ more spokes along random directions, for a total of $3d$ spokes. (One may use more spokes to more accurately estimate the cell size, at the price of higher computational cost.) Each spoke starts infinite, with anchor point c_i, and we trim each end by separating hyperplanes until its end points are on the cell's boundary. If an end point is too close to the domain boundary, we reduce its length so its distance to the nearest boundary plane is at least 1/3 the distance from the center to that plane. We label the end points p_r and p_l. We say the length of the spoke is $\min(\|c_i p_l)\|_2, \|c_i p_r\|_2)$. We approximate the cell size by the longest such length, with spoke with end points l_i and r_i.

Cell refinement. When cell i is chosen for refinement, l_i and r_i are added as new samples, implicitly creating two new Voronoi cells and modifying nearby cells.

Input: sample budget N, function to optimize f
Output: global optimum estimate $f^* \approx \hat{f}$
1: Sample first point x_1, evaluate $f_1 = f(x_1)$
2: Estimate cell size h_1
3: $n \leftarrow 1$
4: **while** $n \leq N$ **do**
5: Construct 2D lower convex hull $\mathcal{H}$ of $\{h_i, f_i\}$
6: **for** each $c_i \in \mathcal{H}$ **do**
7: Refine cell c_i, evaluate f at new points
8: Update cell sizes of new and refined cells
9: $n \leftarrow n + 2$
10: **if** $n \geq N$ **then end**
11: **return** $f^* = \min\{f_i\}$ `// best solution found`

Algorithm 5. Opt-darts

D.3 Analytical experiments

In this section, we used two standard test functions, `Easom` and `Bohachevsky` [Jamil and Yang 2013], over a variable number of dimensions to illustrate the difference between Opt-darts and DIRECT. Two of the test suites which list these functions [Jamil and Yang 2013; Yang 2010] have been collectively cited more than 350 times. They are also available in many online tools and from test function libraries in Matlab [Burkardt 2011; Leong 2016], and R [Bossek 2017]. We chose these two functions to represent two extreme behaviors in the neighborhood of the global minimum $\hat{f}$. The `Easom` function approaches the global minimum via very high gradient. It is almost flat everywhere and has a deep "pinhole" region where the optimum lies. In contrast, the `Bohachevsky` function approaches $\hat{f}$ via an almost flat region that looks like a shallow bowl. Both functions are noisy, and have many local minima. The global minimum $\hat{f} = 0$ for both functions, and is located at the origin, $\forall d$. Note that finding $\hat{f}$ for the class of functions like `Easom` gets significantly harder as dimension increases. This problem is not as significant for the class of functions like `Bohachevsky`. Figure 21 illustrates an informative comparison of DIRECT and Opt-darts in terms of point placement using evaluations of the `Easom` function. In Table 1 in Section 6.2, we compare the number of function evaluations needed to be within 10^{-4} of the true global minimum $\hat{f}$. As shown in the table, opt-darts was able to achieve orders of speedup over DIRECT.

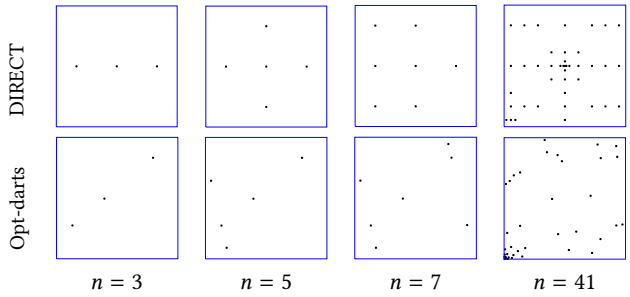


Fig. 21. Contrasting sample placements, over the 2D Easom test function. The global minimum $\hat{f}$ is located at the lower left corner of the domain. Opt-darts approached it much faster than DIRECT.

E APPLICATION: RENDERING

For a traditional rendering demonstration, we have integrated spoke-darts into the Mitsuba physically-based renderer [Jakob 2010]. Most rendering algorithms in Mitsuba use point samplers to generate multidimensional point sets, providing a good base for applying and comparing different sampling methods.

Scenes. We have chosen two scenes for this rendering experiment: Babylon and torus-in-glass, as shown in Figure 22. The Babylon case uses 4d samples corresponding to the 2D camera screen space + the 2D lens space to generate defocus blur. The torus-in-glass cases demonstrates a bidirectional path tracer using 8d samples corresponding to 2D for the sky emitter, 2D for the camera screen, and 2D for each bounce along each camera and light path.

Results. In Figure 22, we compare the rendering quality of our method against the high-quality samplers within Mitsuba: multidimensional stratified sampling, and low-discrepancy sampling based on [Kollig and Keller 2003]. These samplers all seem to be well-spaced only along pairs of dimensions, such as the x-y camera samples and the u-v lens samples, but not the joint domains in

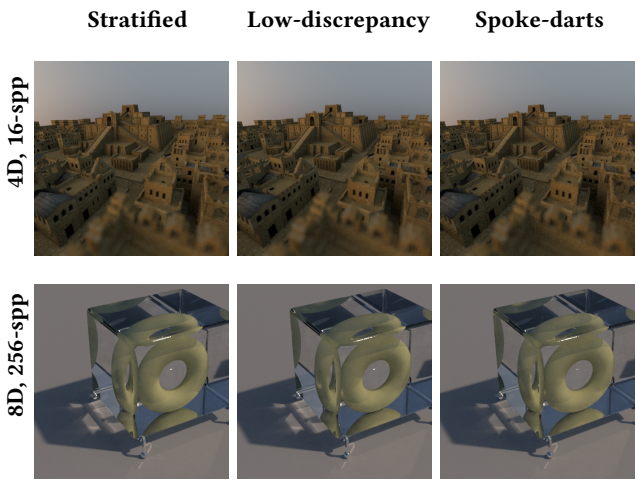


Fig. 22. Rendering by Mitsuba using two of the default Mitsuba samplers and Spoke-darts. The Babylon scene (top) renders antialiased depth-of-field using 16 samples-per-pixel (spp) in 4 dimensions, and the torus-in-glass scene (bottom) renders antialiased using 256-spp in 8D.

higher dimensions such as the 4D camera + lens domain. In contrast, spoke-darts samples are well-spaced along the joint domains. In Figure 22, the rendering quality using spoke-darts is comparable to that of using Mitsuba's samplers. As analyzed in [Reinert et al. 2016], the rendering quality depends on the projected sample distributions, which might explain the comparable quality of Mitsuba samplers and our method. Nevertheless, our method guarantees good sample distributions in any dimensions and performs at least as well in projected dimensions, even without explicit consideration of projected distributions as in [Reinert et al. 2016].

F APPLICATION: MOTION PLANNING

Motion planning algorithms are frequently used in robotics, gaming, CAD/CAM, and animation [Overmars 2005; Pan et al. 2010; Yamane et al. 2004]. 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, such as path smoothness, dynamics constraints, and plausible motions. This problem has been extensively studied for more than three decades. Two main challenges are:

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

High 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.

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.

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. Park et al. [2016] demonstrated that using Poisson-disk sampling instead can lead to more efficient exploration of the configuration space. We summarize [Park et al. 2016] in Algorithm 6. However, the algorithm described in [Park et al. 2016] assumes availability of precomputed Poisson-disk samples that can guide the selection of new points which are not too close to prior points. It starts with uniform sampling in the high dimensional space, and generates more adaptive samples in tight space or narrow passages. Furthermore, the Poisson-disk sampling can be used to design a parallel version of RRT algorithm that can map well to current commodity processors, including multi-core CPUs and many-core GPUs. Using a precomputed sampling that is shared by all threads allows efficient detection when a tree branch reaches an area that is already explored, and avoids redundant exploration.

Input: configurations $\mathbf{x}_{init}$ and $\mathbf{x}_{goal}$ in domain Ω
Input: Poisson-disk sample set $\mathbf{P}$ precomputed via Algorithm 1
Output: RRT Tree $\mathbf{T}$

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

Algorithm 6. Parallel Poisson-RRT with precomputed samples.

We use the novel spoke-darts algorithm to precompute the high-dimension sample set via spoke-dart sampling and also adaptively refine this set to compute collision-free paths through narrow passages. This high-dimensional set is used by the resulting Poisson-RRT based motion planning algorithm [Park et al. 2016]. Furthermore, it is used to design a practical parallel RRT in high-dimensional configurations space, e.g., for a 23 DOF robot. We highlight the performance of this novel parallel Poisson RRT planner on three well-known motion planning benchmark scenarios from OMPL [Şucan et al. 2012]. 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 1d. The total times taken by the *planner* are shown in Table 2.