Fast generation of centroids for MAP-Elites

Jean-Baptiste Mouret

Fast generation of centroids for MAP-Elites

Jean-Baptiste Mouret
Inria Nancy – Grand Est, CNRS, Université de Lorraine
Nancy, France
jean-baptiste.mouret@inria.fr

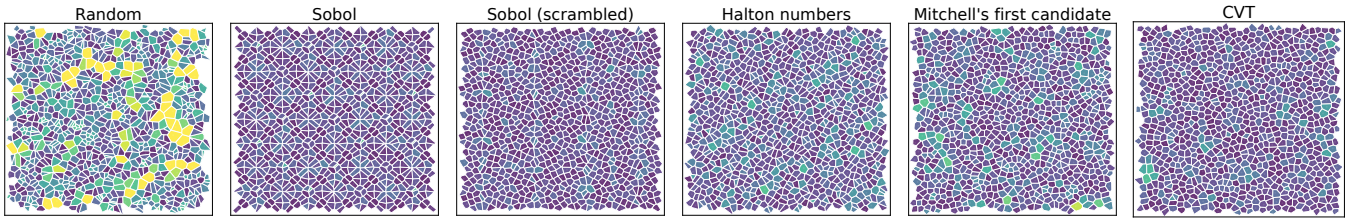

Figure 1: 2-dimensional Voronoi diagrams from 1024 centroids generated with 5 different algorithms. The color of each cell represents its volume difference from the optimal split (1/1024), where yellow indicates the maximum difference and dark violet indicates the minimum. In this 2-dimensional example, the pseudo-random generation ("Random") leads to regions with more variance in surface compared to the other algorithms. See Fig. 2 for quantitative results in higher dimensions.

ABSTRACT

The use of MAP-Elites in high-dimensional behavioral spaces requires a scalable method for dividing the space into regions of equal volume. So far, the recommended approach to generate these regions has been the Centroidal Voronoi Tesselation (CVT), but this algorithm has a significant computational cost (typically a few minutes for more than 50 dimensions). In this paper, we investigate alternative approaches to generate regions of equal volumes for MAP-Elites. In particular, we experiment with generating region centroids with low-discrepancy sequences (Sobol, Halton), pseudo-random numbers, and a simple blue noise generator. Our results show that, for spaces with 100 dimensions, most methods perform similarly, including pseudo-random numbers. For spaces with dimensions between 5 and 50, a CVT generates significantly better centroids. In lower dimensions (1-5), a scrambled Sobol sequence generates well-spread centroids in a few milliseconds.

CCS CONCEPTS

• **Theory of computation → Evolutionary algorithms**.

KEYWORDS

MAP-Elites, centroids, low-discrepancy

ACM Reference Format:
Jean-Baptiste Mouret. 2023. Fast generation of centroids for MAP-Elites. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583133.3590726

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal
2023. ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3590726

1 INTRODUCTION

The MAP-Elites [10] algorithm searches for high-performing solutions that exhibit diverse behaviors, that is, occupy distinct behavioral "niches". In recent years, MAP-Elites has been successfully used in a wide range of domains, including robotics [2, 3], nanomaterials [7], and video game levels [8].

The original version of MAP-Elites [10] divides the behavioral space with a regular grid (e.g., a 100×100 grid). This approach works well for low-dimensional spaces, as a 100×100 2-dimensional grid results in $10,000$ niches, and a 20×20×20 grid results in $8,000$ niches. However, this grid-based division of space becomes impractical in higher dimensions. For example, in a 20-dimensional space, dividing each dimension in only 2 values results in more than 1 million cells (2^{20}) which is likely to take an excessive amount of time to fill. Furthermore, using a grid to divide the space ties the number of niches to the dimensionality of the space, whereas the number of niches should be defined by the desired results. For instance, in 10-d, a regular grid can only be made of 1024 (2^{10}), $59,049$ (3^{10}), or more than 1 million cells, but a user could wish to fill 5000 cells.

CVT-MAP-Elites [14] addresses this problem by performing a Centroidal Voronoi Tesselation (CVT) of the behavioral space (usually a hypercube), which converges to a partition of the space into cells of equal volume. Each cell is defined by a centroid, that is, a point in the behavioral space, and the corresponding cell consists of all points closer to that centroid than to any other. This partition is called a Voronoi tesselation (or a Voronoi diagram). CVT-MAP-Elites makes it possible to generalize the grid-based MAP-Elites to any dimension by implementing a simple modification: instead of using a discretization to find the cell given a behavioral descriptor, the algorithm uses a nearest neighbor search to find the closest centroid. The rest of the MAP-Elites algorithm stays exactly the same as for a grid.

In the original CVT-MAP-Elites article [14], the authors found well-spread centroids (and therefore equal volumes for each cell)

using the McQueen's algorithm [5], which is essentially the K-means algorithm on a large number of random points, with the number of cluster equals to the number of centroids. However, this computation can require several minutes (even hours), especially on a high number of centroids (e.g. 30,000) in high-dimensional spaces (>20). In addition, the K-means algorithm is simple in principle, but it exists in many variants and a state-of-the-art implementation is not trivial; arguably, K-means can be more complex to implement than MAP-Elites, which means that adding K-means or a Lloyd relaxation [4] to MAP-Elites makes the latter substantially more complicated.

In this paper, we investigate faster algorithms to generate a set of well-spread centroids that can be used with CVT-MAP-Elites, thus making it possible to scale-up MAP-Elites to arbitrary behavioral dimensions without requiring to compute a CVT.

2 FAST ALGORITHMS FOR WELL SPREAD POINTS

Our first idea is to use a pseudo-random number generator to sample points and use them as centroids. However, pseudo-random number generators are primarily designed to make the next point hard to predict, not to cover the space evenly when only a few samples are drawn. In fact, if the generator were to cover the space well, the next pseudo-random point would be easy to predict: it should be in the least dense region. Ultimately, over a large number of draws, a good pseudo-random number generator should be unbiased, meaning that the values should be uniformly distributed. However, this property is not true when a low number of samples (thousands) is used, which is the case when generating centroids.

Low-discrepancy sequences are an alternative to pseudo-random number generators that are designed to cover the space as well as possible when new samples are drawn [1, 11, 13]. They are often used in Monte Carlo integration and to initialize optimization algorithms (e.g., in Bayesian optimization). Many sequences have been proposed and are available in common mathematics packages; the most common ones, from the sixties, are Halton numbers [6] and Sobol numbers [13]. These sequences have a low computational cost (a few arithmetic operations), but they often rely on a list of specific numbers (e.g., prime numbers) that are computed beforehand. When the sequence does not need to be deterministic, "scrambling" the generated numbers [1, 11], which is typically achieved with bit operations that do not change the discrepancy, can substantially improve the result while keeping the computational cost low.

A third family of point generation algorithms is blue noise generators, like Mitchell's first candidate algorithm [9]. While standard random number generators produce "white noise", blue noise generators aim at creating a sequence of random points that are distributed such that no two points are too close to each other, which is term as "low spatial correlation" or "low-frequency noise[1]." Blue noise generators are often aimed at Monte-Carlo ray tracing/path tracing algorithms, and most algorithms have been published in the computer graphics field.

[1]The term "blue noise" comes from the fact that the power spectrum of the point distribution has a higher energy at higher frequencies, similar to the blue end of the visible light spectrum.

3 COMPARING ALGORITHMS

Our objective is to obtain a set of points that evenly partition the hypervolume. One way to compute the volume of each cell is to compute the Voronoi diagram, then the convex hull of each polygon. Unfortunately, the commonly available algorithms to explicitly compute a Voronoi diagram do not scale well to high-dimensional spaces, which is important for our study. To keep things simple to implement, scalable, and bounded in computation time, at the expense of some loss in precision, we chose to assess the volume of each cell using a Monte Carlo algorithm that evaluates the probability of sampling a point in each region. The best division of space should yield an equal probability for each cell. The Monte Carlo algorithm works as follows:

(1) let C be the set of $|C|$ centroids for which we want to evaluate the volume;
(2) generate N random numbers[2] (here we use 1 million points);
(3) for each sample, find the closest centroid[3]
(4) for each centroid c, count how many samples have selected c as the closest centroids;
(5) we consider that the volume of each region is the number of points sampled in that region normalized by the total number of points (that is, the probability of sampling a point in that region with a pseudo-random number generator).

In a mathematical form:

$$p(c) = \frac{1}{|S|} \sum_{s \in S} h(s, c) \tag{1}$$

where:

$$h(s, c) = \begin{cases} 1 \text{ if } c = \arg\min_{x \in C} \left\| s - x \right\|^2 \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

To get a score, we compare the probability $P(c)$ of "hitting" a region c with the theoretical probability with equal volumes ($1/|C|$):

$$V(C) = \frac{1}{N} \sum_{c \in C} \left| p(c) - \frac{1}{|C|} \right| \tag{3}$$

4 ALGORITHMS

We compare 6 algorithms to sample centroids, using Numpy's and Scipy's implementations.

(1) Random centroids (we use Numpy's random number generator (PCG64));
(2) Sobol sequence [13] (*scipy.stats.qmc*, or boost.random in C++)
(3) Scrambled Sobol sequence [1, 11] (*scipy.stats.qmc*)
(4) Halton numbers [6] (*scipy.stats.qmc*)
(5) Mitchell's first candidate algorithms [9], which is a simple blue noise generator that iteratively (1) randomly generates k candidates (e.g., 10), (2) computes the distance to all the

[2]Ironically, using a low-discrepancy sequence instead of random numbers would make it possible to use fewer samples for the same quality of the estimated quantity, but we decided to use a standard random number generator to avoid the potential correlations between the sampling algorithm and the centroids.

[3]Using a spatial tree could make this nearest neighbor search fast by avoiding computing the full distance matrix, but spatial trees are often not effective in more than 10-20 dimensions. By contrast, computing the full distance matrix is theoretically more expensive but it can easily be vectorized.

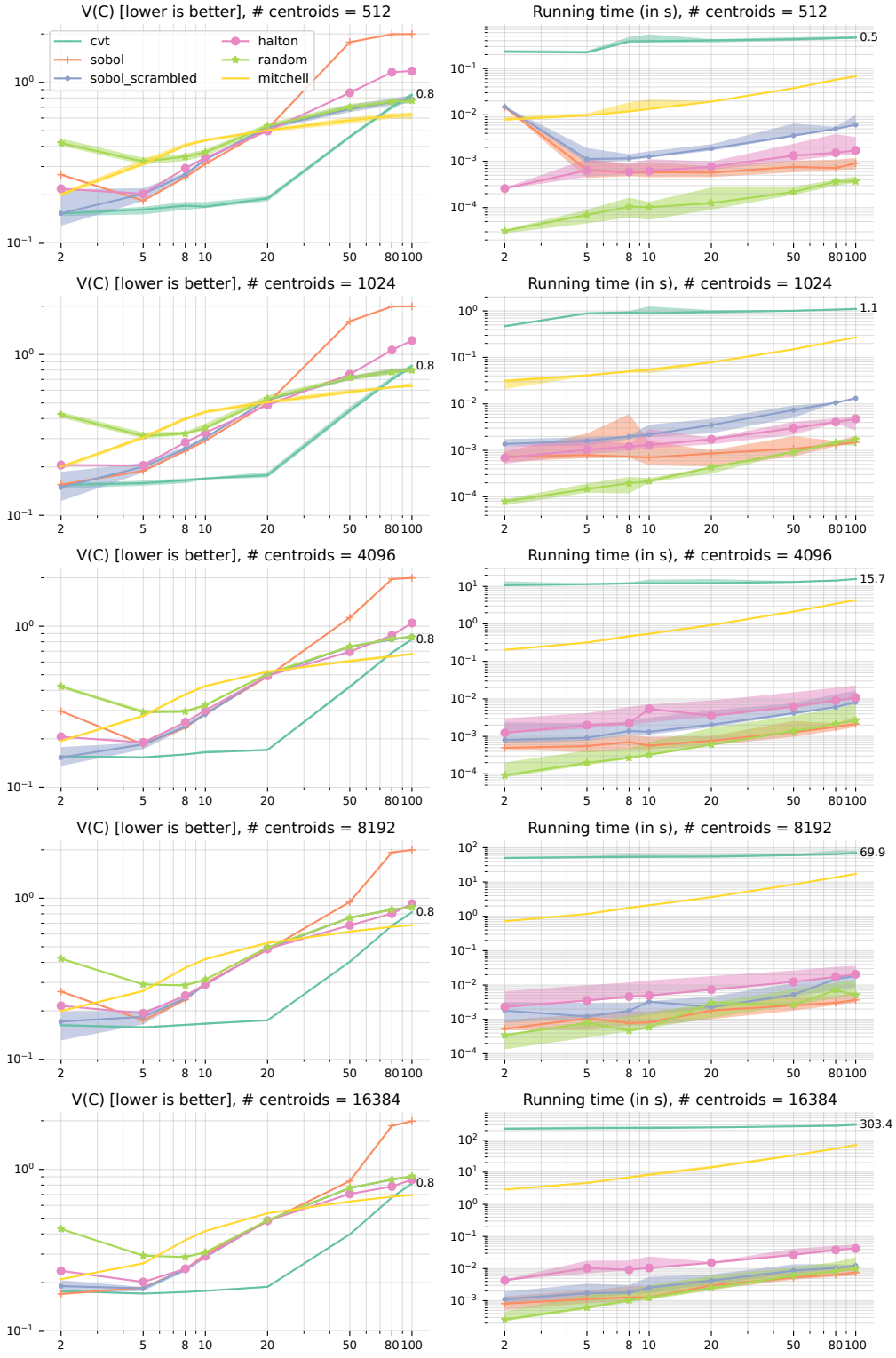

Figure 2: (left). $V(C)$ **for dimensions 2, 5, 10, 50 and 100, and number of centroids from 1024 to 16384 (lower is better). (right). Running time (in seconds, lower is better). The lines represent the median over 10 independent replicates and the shaded zone the** $(5\%, 95\%)$ **confidence interval. Most of the uncertainty is not visible of the plots because it is too small. The number on the top right corresponds to the median of the CVT method for 100 centroids.**

centroids so far, (3) selects as a new centroid the candidate that has the highest mean distance to the current centroids;
(6) Centroidal Voronoi Tesselation [4], like in CVT-MAP-Elites [14] (*sklearn.cluster* for K-means).

We compare the $V(C)$ score with dimensions from 2 to 100 and from 1024 to 16,384 centroids.

5 RESULTS

In two dimensions (Fig. 1), the random generation of centroids gives the most irregular results, although it could be sufficient for many applications. The Sobol centroids are well spread, but they are organized in a visible "star-like" pattern. This effect mostly disappears when the Sobol numbers are scrambled, which gives a visually good result. Even without scrambling, Halton numbers lead to well-spread centroids, which are not visually worse than those obtained with Mitchell's first candidate algorithms and the CVT.

When the number of centroids is changed (Fig. 2), the behavior of each algorithm does not change significantly. The variance of the results is very low (invisible on the plot for deterministic method), which shows that enough samples were used for the Monte-Carlo estimate (section 3).

The dimensionality of the centroid space has a large influence on the score (Fig. 2, left). The CVT obtains the best score for all the dimensions, but it gets worse after 50 dimensions, up to being equivalent to random number generations in dimension 100. This might come from the fact that we would need to sample more points before the clustering step (which would substantially increase the computational cost) when the dimension increases. The Sobol numbers lead to good centroids up to 10 dimensions, but with more than 10 dimensions they become worse than a standard number generator. Halton numbers follow a similar pattern: they lead to good centroids in low-dimension but worse than a random number generation in high-dimension (here, > 50). This is a well-known issue with Sobol numbers and the main rationale for scrambling [12]. The scrambling helps significantly and prevents the Sobol numbers to lead to worse centroids than random generation in high-dimensional space, but this is not enough to create better centroids. Mitchell's first candidate algorithm gives slightly better results than the low-discrepancy methods in high-dimensions, and good results in low-dimensions (2-3); but in dimensions 5 to 50, it leads to worse results than the low-discrepancy methods.

In term of running time, the low-discrepancy methods have a very low run-time (e.g., about 10 ms to generate 16384 100-dimensional centroids with the scrambled Sobol sequence). The CVT has a much higher running time that grows with both the number of centroids and the dimensionality, from about 20 ms in 2-D for 512 centroids, to about 300 s (5 minutes, on a modern, multi-core computer) for 16384 centroids in a 100-dimensional space. Mitchell's first candidate algorithm has a lower computational cost in our implementation (which bounds the number of candidates by a constant), but can still take more than 1 minute to generate 16384 100-dimensional centroids. Please note that this algorithm could be made much faster in low dimensions by using a spatial tree, but spatial trees are most of the time ineffective in high-dimensions.

6 CONCLUSION

- If running time is not an issue (e.g., for very long fitness evaluations), the CVT gives the best results.
- For more than 10 dimensions, the basic pseudo-random numbers generators are as good as low-discrepancy sequences (Sobol, Halton) and are available in any programming language.
- For less than 10 dimensions, the scrambled Sobol sequence is the best algorithm (except CVT), closely followed by unscrambled Halton numbers.

Overall, the MAP-Elites algorithm does not require the best division of space, but a better division of space will yield to better spread elites. In our opinion, scrambled Sobol numbers are a fast alternative to the CVT with well-studied properties and implementations available in common Python and C++ libraries (scipy, boost), but pseudo-random numbers are a good option for high-dimensional spaces. In future work, we will investigate how to provide a large database of precomputed high-quality tesselations that can quickly provide centroids in any programming language: at this time, this seems to be the best approach to get both high-quality centroids and a fast running time.

ACKNOWLEDGMENTS

This project is supported by the Direction General de l'Armement and the EurROBIN Horizon project (grant number 101070596).

REFERENCES

[1] Brent Burley. 2020. Practical hash-based Owen scrambling. *Journal of Computer Graphics Techniques (JCGT)* 10, 4 (2020), 29.
[2] Konstantinos Chatzilygeroudis, Vassilis Vassiliades, and Jean-Baptiste Mouret. 2018. Reset-free trial-and-error learning for robot damage recovery. *Robotics and Autonomous Systems* 100 (2018), 236–250.
[3] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* 521, 7553 (2015), 503–507.
[4] Qiang Du, Maria Emelianenko, and Lili Ju. 2006. Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM journal on numerical analysis* 44, 1 (2006), 102–119.
[5] Qiang Du and Tak-Win Wong. 2002. Numerical studies of MacQueen's k-means algorithm for computing the centroidal Voronoi tessellations. *Computers & Mathematics with Applications* 44, 3-4 (2002), 511–523.
[6] John H Halton. 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* 2 (1960), 84–90.
[7] Yibin Jiang, Daniel Salley, Abhishek Sharma, Graham Keenan, Margaret Mullin, and Leroy Cronin. 2022. An artificial intelligence enabled chemical synthesis robot for exploration and optimization of nanomaterials. *Science Advances* 8, 40 (2022), eabo2626.
[8] Ahmed Khalifa, Scott Lee, Andy Nealen, and Julian Togelius. 2018. Talakat: Bullet hell generation through constrained map-elites. In *Proceedings of The Genetic and Evolutionary Computation Conference*. 1047–1054.
[9] Don P Mitchell. 1991. Spectrally optimal sampling for distribution ray tracing. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. 157–164.
[10] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).
[11] Art B Owen. 1995. Randomly permuted (t, m, s)-nets and (t, s)-sequences. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing: Proceedings of a conference at the University of Nevada, Las Vegas, Nevada*. Springer, 299–317.
[12] Loïs Paulin, David Coeurjolly, Jean-Claude Iehl, Nicolas Bonneel, Alexander Keller, and Victor Ostromoukhov. 2021. Cascaded Sobol'Sampling. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–13.
[13] Ilya Meerovich Sobol'. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 7, 4 (1967), 784–802.
[14] Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2017. Using centroidal Voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation* 22, 4 (2017), 623–630.