Conway Crossover to Create Hyperdimensional Point Packings, with Applications

Daniel Ashlock, Senior member IEEE, Steffen Graether

Abstract—Point packings in the unit square are placements of \( n \) points in the unit square that maximize the minimum distance between any two of the points. Such packings are surrogates for the 2D-stock cutting problem. In this study we examine a unique representation for the point packing problem and extend the problem to higher dimensions and more complex shapes. The representation uses the Conway operator, a \( k \)-ary variation operator based on the lexicode algorithm. Three application of point packings are demonstrated. A parameter study for the Conway operator based algorithm is performed demonstrating that large populations are uniformly desirable but that the part of the operator that corresponds to mutation has a strongly problem dependent value for good performance. The three applications demonstrated are selecting well-space RGB color palettes, initialization of populations in an evolutionary optimizer, and fast clustering of codon usage data. Color palettes of different size are presented. The optimization application is found to gain substantial performance by using point packings as initializers. The bioinformatics application demonstrates significantly non-random clustering of a family of intrinsically disordered proteins known as dehydrins.

I. INTRODUCTION

Finding a collection of \( n \) points in the unit square that maximize the minimum distance between any two points [7] is a surrogate for the two-dimensional stock cutting problem. This consists of placing items on a sheet of material to be stamped or cut as efficiently as possible. This problem is an area of active research in computational intelligence and adaptive computation [4], [19], [5].

Definition 1: The point packing problem in a closed set \( S \) or \( \text{PPS problem} \) seeks to place a specified number of points into the set \( S \), including its boundary, so that the minimum distance between any pair of points is maximized.

A web repository \(^1\) gives the current best known values for the point packing problem in the case that \( S \) is the unit square. The distance column of the table at this web site contains the best known fitness. In this study we examine the dual problem of packing as many points into a set \( S \) as possible, given a specified minimum distance \( \delta \). In [2] two representations for evolving solutions to the point packing problem were compared. This earlier study used the \( n = 8 \) point version of the problem to compare the representations.

This small problem size was used for the comparison because the representations used in the earlier study did not scale efficiently. A new representation, based on an algorithm for finding embeddable DNA biomarkers [3], is introduced in this study to address the scaling problem.

The key to the new representation is the Conway operator. This operator can operate on one or more point packings to produce new ones. The operator is described in detail in Section III. In [3] this operator was used to build error correcting codes over the DNA alphabet using the \text{Levenshtein} or edit metric. The words making up such an error correcting code are useful as tags in DNA constructs because they are a large number of sequencing errors apart. This means that, once constructs are sequenced, the tag can still be identified in many cases. In [14] a 50% recovery of sequence identities that did not exhibit clean tags was obtained. The algorithm used to create DNA tags works for any notion of “well spaced out” and for any metric. This means that re-purposing the Conway operator to locate point packing is not difficult.

The remainder of this paper is structured as follows. Section II examines past work on the point packing problem. In section III we specify the Conway operator and the representation for evolving point packings based on it. Section IV gives the experimental design. Section V gives and discusses results, including reporting the results of applications of point packings. Finally, Section VI draws conclusions and outlines possible next steps.

II. BACKGROUND

The problem of packing points into the unit square (or hypercube) is equivalent to the problem of packing circles of a uniform radius, by simply rescaling the set that is the target of the packing. Figure 1 shows an example of the rescaling needed to transform the point problem into the circle packing problem. Effectively, the square is rescaled so that point centers on the boundary of the unit square become the centers of circles tangent to the unit square. With this equivalence we will simply treat literature on both problems, point packing and circle packing, as being about the same problem.

The first problem involving point packing in the unit square was solved in [15], treating the problem of packing eight points into the unit square. A large number of particular solutions for specific numbers of points appeared in the literature thereafter. A typical approach is to find circle-packings that arise from geometrically regular configurations [11]. A survey of results up until 2007 appears in [18] together with a historical

---

Daniel Ashlock is with the Department of Mathematics and Statistics at the University of Guelph, dashlock@uoguelph.ca

Steffen Graether is with the Department of Molecular and Cellular Biology at the University of Guelph, graether@uoguelph.ca

The authors thank the University of Guelph, and the Canadian Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting this work.

\(^1\)http://hydra.nat.uni-magdeburg.de/packing/csq/csq.html
A new set of points with minimum distance random order. Algorithm 1 is then applied to this set to retrieve \( R \) additional points generated at random are shuffled into a collection \( \{ \} \) that plays the same role as mutation. To apply the CVO to a collection \( \{ C_1, C_2, \ldots, C_k \} \) of sets of points the sets and \( R \) additional points generated at random are shuffled into a random order. Algorithm 1 is then applied to this set to retrieve a new set of points with minimum distance \( \delta \) or more. The order of points matters for application of the CVO, but it is established during application of the CVO, and so both the input sets and the output set may be viewed as unordered sets of points.

In the past, this operator has been used in both unary and binary form to create sequencing-error tolerant labels over the DNA alphabet. This study uses the operator to evolve point packings into a variety of sets. These applications are quite similar as both amount to evolving an error correcting code over some metric space - the edit space of strings and Euclidean space, respectively.

\textit{Algorithm 1: Conway's Lexicode Algorithm}  
\textbf{Input:} an set \( S \) of points in some order. \( a \) minimum distance \( \delta \). \n\textbf{Output:} a subset \( T \) of \( S \) with minimum distance \( \delta \). \n\textbf{Details:}  
\begin{enumerate}  
\item Initialize \( T \) to be empty.  
\item Traversing \( S \) in order,  
\item Add a point from \( S \) to \( T \) if its distance  
\item from the current members of \( T \) is at least \( \delta \).  
\end{enumerate}  
\textbf{Return}(\( T \))

This study examines a variety of sets and minimum distance parameters as well as performing a parameter study on a CVO-based evolutionary algorithm. The CVO based algorithm was chosen from several possible representations for two reasons. First, in preliminary testing it exhibited superior performance when \( S \) was \([0,1]^6\), the six-dimensional unit hypercube with \( \delta = 0.5 \). The case of the PPS problem has been selected for performance of the parameter study. Second, it is exceedingly easy to control the CVO based algorithm to operate on different choices of \( S \), a property that other representations did not share.

Any algorithm that moves individual points must constantly check to see if the operator that move points keeps them within the set \( S \). The CVO never moves points. This means that as long as the random point generator used for creating the initial population, and for random material addition, only generates points within \( S \), the algorithm will always select points inside of \( S \). This avoids possibly expensive repair code or the need to design abstruse mutation operators that intrinsically respect the boundaries of \( S \). As an example, one of the application requires points in \( \mathbb{R}^{64} \) that obey the constraint \( \sum x_i = 1 \).

This constraint is easy to incorporate into the random point generator.

\textbf{A. Mathematical Properties of Hypercube Packings}  
This section gives a handy mathematical result that lets us build very large point packings from smaller ones in lower dimensions. This result is that, as long as we are packing points into a hypercube, the size of the best packing is multiplicative across dimensions.

\textbf{Definition 2:} If \( S = [0,1]^n \) is the unit hypercube in \( n \)-dimensions then \( f_\delta(n) \) is the largest number of points that can be packed into \( S \).

\textbf{Theorem 1:} If \( n = m + k \) for positive integers \( n, m, k \), then  
\[ f_\delta(n) \geq f_\delta(m) \cdot f_\delta(k) \]

\textbf{Proof:}
Let \( C_1 \) be a set of points in \( \mathbb{R}^m \) that is a witness to \( f_{\delta}(m) \) and let \( C_2 \) be a set of points in \( \mathbb{R}^k \) that is a witness to \( f_{\delta}(k) \). Compose the set \( T \) in \( \mathbb{R}^n \) by taking the cross product \( T = C_1 \times C_2 \). Then \( T \) is composed of all points whose first \( m \) coordinates are a point in \( C_1 \) and whose last \( k \) coordinates are a point in \( C_2 \). Any pair of points in this set is at distance at least \( \delta \) because the distance is at least \( \delta \) for at least one of the first \( m \) and last \( k \) coordinates. It follows the \( T \) has the required minimum distance. The size of the set is \( |T| = |C_1||C_2| \) and the theorem is complete. □

As we will see there is a packing with 21 points in \( H_3 = [0, 1]^3 \) which implies the existence of a packing with 441 points in \( H_6 = [0, 1]^6 \), which is slightly better than the directly evolved packings for \( H_6 \) in the experiments. Such a product packing is also more geometrically regular which is a bad property for the population initialization application.

We now turn to a result that lets us use one hypercuboidal point packing in a large number of ways. The group of symmetries of the hypercube are called the hyperoctehedral group [6]. Since these symmetries are isometries (distance preserving maps) it follows that if we apply a symmetry of the hypercube to a point packing in the hypercube then we obtain a different point packing with the same distances between points. These symmetries can all have the following form. First, choose a permutation of the coordinates of the points - this permutes the axes of \( \mathbb{R}^n \) in a manner that takes the unit hypercube to itself. After this, for some subset of the coordinate positions, apply the map \( x \rightarrow (1 - x) \) to those coordinates. This operation swaps the opposite faces of the cube in the dimensions corresponding to the selected coordinates.

Hyperoctehedral symmetries are used in one of the applications given in Section V. The point packings are used as initial populations for high dimensional optimization; the hyperoctehedral symmetries each yield a different initial population. The number of such symmetries in \( n \) dimensions is \( 2^n \cdot n! \). As the number of dimensions increases this provides a generous collection of initial populations.

IV. EXPERIMENTAL DESIGN

The parameters needed for an evolutionary algorithm that evolves packings of points into a set \( S \) are the population size \( Z \), the random material rate \( R \) (defined above), the amount of time evolution will be run, and the number \( K \) of sets fed into the Conway variation operator. The evolutionary algorithm we use is based on size seven single tournament selection using the CVO. The default parameters are \( Z = 4000 \), \( R = 20 \), and \( K = 2 \). Population members are initialized by generating 1000 random points and then applying Algorithm 1 to those points. An instance of the point packing problem requires that the set \( S \) and the minimum distance \( \delta \) be specified. Fitness of a point packing is the number of points it contains.

A parameter study is performed on the six dimensional unit hypercube with minimum distance \( \delta = 0.5 \) for the following parameter values. Population size is varied across \( Z = \{10, 100, 1000, 4000\} \), the random material rate is varied across \( R = \{10, 20, 30, 40\} \), and the degree of the Conway operator is varied across \( K = \{1, 2\} \).

Experiments are run with the parameter values located in the first set of experiments for \( S = [0, 1]^3 \), the unit cube, with \( \delta \in \{0.25, 0.375, 0.5, 0.625\} \). These experiments are used for the application of locating well-space sets of colors. The coordinates of points in \( \mathbb{R}^3 \) are mapped onto RGB values.

In support of a bioinformatics application, the clustering of codon usage profiles, a series of experiments were run for the set \( S = \{x_{64}^{64} : \sum x_i = 1\} \) the 63-dimensional simplex whose vertices are the standard basis of \( \mathbb{R}^{64} \). Trial and error was used to determine that \( \delta = 0.12 \) yields codes with just over 100 points, a reasonable number for the application. These experiments used a population size of \( Z = 400 \), random material rates of \( R = \{10, 20, 30, 40\} \), and \( K = 2 \). The smaller population size was needed to prevent the algorithm from slowing down too much on the hardware used. The parameter \( Z \) was thus set “as large as practical”. Earlier experiments demonstrated that \( K = 2 \) is a superior value. The random material rate has already yielded different behaviors for different problems and so several values \( R = \{10, 20, 30, 40, 50\} \) were tried.

To enable the optimization application, three additional experiments were performed. One of the experiment on \( S = [0, 1]^6 \) with \( \delta = 0.5 \) yielded a packing with 411 points. Additional experiments on the hypercubes of dimension 3, 9, and 12 were run with \( \delta = 0.13, 0.8 \), and 1.1 yielding point packings with 408, 417, and 412 points respectively. These packings were used as population initializers.

V. RESULTS AND DISCUSSION

Figure 2 shows the comparison of different population sizes for the six-hypercube, \( \delta = 0.5 \) case of the point packing
Fig. 3. Shown are wasp plots comparing the performance for different random material addition rates for the $\delta = 0.5$, $S = [0, 1]^6$ version of the packing problem.

There is a statistically significant and clear benefit to larger population sizes. The size of 4000 was used, instead of 10,000, because of problems with running time with populations of 10,000 population members. Results for the random material rate are less clear with $R = 10$ having a slightly better average than $R = 20$, but with $R = 20$ the extreme values are better. The rates higher that $R = 20$ were clearly inferior. The lack of a simple pattern for this experiment let to comparing random material rates on two different cases of the packing problem.

Fig. 4. Shown are wasp plots for the $K = 1$ version of the algorithm for the $\delta = 0.5$, $S = [0, 1]^6$ version of the packing problem with different random material addition rates.

Figure 4 shows the results for the unary version of the CVO. Contrasting Figure 2 with Figure 4 we also see that the response to changing the random material rate is different for the unary and binary versions of the CVO. This suggests that the way the two algorithms search the space of point packings is very different. The failure of the unary version of the CVO motivated the use of the binary version for the experiments one the 3-cube and the 63-simplex of points in 64 dimensions whose coordinates sum to one.

Figure 5 shows the results of the experiments evolving packings in the 3-cube. The applications section shows how these packings are transformed into well-spaced sets of RGB colors. The 3-cube is a relatively easy case of the packing problem for the five minimum distance parameters chosen. Given the equivalence of point packings and sphere packings, the number of points that can be packed into a 3-cube should grow roughly as the cube of the reciprocal of the minimum distance parameter.

Figure 6 shows the result of changing the random material rate when the algorithm is packing points into the simplex of constant sum one inside $[0, 1]^{64}$. This experiment represents a third behavior. When packing points into $[0, 1]^6$ with the binary CVO, an intermediate random material rate was best; the unary CVO for packing points into $[0, 1]^6$ worked best with the lowest random material rate. When packing points into the high dimensional simplex, the highest random material rate was best. The degree of improvement for increasing $R$ was statistically significant with a best result of 127 points when $R = 50$. The trend is upward suggesting that even better results may be possible for higher material addition rates.

A. Applications

The evolved point packings will be used in three applications. Finding well spaced RBG colors, initialization of popu-
B. RGB Colorings

The point packings in $[0,1]^3$ can be mapped into RGB colorings by letting each coordinate specify the intensity of one of the color channels. Figure 7 shows a maximum fitness packings, rendered as selections of colors, sorted by Euclidean distance of the points from (0,0,0) (black). The best result from thirty runs for each of the four largest values of $\delta$.

The HSV representation for colors makes it easy to find contrasting colors by selecting colors based on equally spaced values of the periodic hue value, but this technique breaks down as the number of distinct colors required increases. More complex cylindrical meshes also using the saturation and value parameters are possible and still easier than hand selection of contrasting RGB values.

The renderings of point packings as RGB colors in Figure 7 were created to be well spaced out. There is an apparent periodicity in the hue in some of the sets of colors - this is an artifact of the geometric regularity of the packing relative to the distance of the individual colors from (0,0,0) (black).

C. Optimization Results

The second application we want to examine is the impact of using point packing as initial populations for evolutionary optimization. A simple evolutionary optimizer was used with three sorts of population initialization: uniformly at random, using a fixed point packing as the initial population, and using a randomly selected hyperoctahedral symmetry of a particular point packing as an initial population - a different symmetry in each run of the evolutionary algorithm. The function being optimized is a constructively multimodal function given by:

$$f(x_1, x_2, \ldots, x_n) = \sum_{k=1}^{n} \frac{x_k}{40} + \cos(x_k), \quad 0 \leq x_i \leq 100.0 \quad (1)$$

Sets of 30 runs were performed for $n = 3, 6, 9,$ and 12 dimensions. The number of distinct maxima are approximately 4000 in 3 dimensions, 16 million in six dimensions, 6.55e10 in nine dimensions, and 2.64e14 in twelve dimension. The algorithm was run until it hit a fitness value within 95% of the global optima. The log of the time to solution is used to compare algorithm performance.

The simple evolutionary algorithm used population sizes of 408, 411, 417, and 412 points in 3, 6, 9, and 12 dimensions respectively. These population sizes represent the largest packing found in the hypercube $[0,1]^n$ experiments run to find point packings for population initialization. In order to initialize populations in the domain $[0,100]^n$ the point packings were scaled by a factor of 100.

The simple evolutionary algorithm used to compare different population initializations was steady state [17] using size four tournament selection. This model of evolution selects four population members, copies the two most fit over the two least fit, and performed two-point crossover of the copies followed by mutation. The mutation operator selects 1-3 genetic loci and adds a Gaussian random variable with mean $\mu = 0$ and standard deviation $\sigma = 2.0$ to the selected position. The standard deviation was tuned to permit an intermediate level of mutational progress. When a value left the permitted boundaries it was mapped to the nearest in-bounds value. A collection of thirty runs were performed for each number of
Fig. 8. Results of the population initialization studies in 3, 6, 9, and 12 dimensions. Three population initialization techniques were compared, random, use of a single point packing, and use of randomly selected hyperoctahedral symmetries of the point packing. Note that the vertical scale is different for $n = 3$ dimensions.

In all four test cases, raw and hyperoctahedral initialization exhibited performance significantly superior to standard random initialization. It is worth noting that the vertical scale of Figure 8 is logarithmic. The degree of advantage generally increased with dimension. The use of the hyperoctahedral permutation of the point packing used for initialization did not yield a significant advantage in overall performance, but it did increase the variability of the outcomes. This means that if finding a better optima is critical, the hyperoctahedral initialization casts a broader net for good optima. The number of different possible hyperoctahedral initializations in $n$ dimensions is $2^n \cdot n!$, yielding 48 in 3 dimensions, 46,080 in six dimensions, 185,794,560 in nine dimensions and 1,961,990,553,600 in twelve dimensions. While each symmetry of the initial population samples the search space in a different way, 30 trials is a meager sample for all but the three-dimensional case.

D. Clustering Codon Use Records

This application uses the packings into the 63-dimensional hyperplane of points with positive coordinates summing to one to perform extremely fast clustering of points that naturally lie within that hyperplane. The clustering algorithm is not sophisticated - data records are binned by the point in the packing they are closest to.

The code of life or genetic code specifies the way 64
possibly 3-base codons over the DNA alphabet \{C, G, A, T\} can specify each of 20 amino acids and codons that mark the end of a genetic specification of a protein. This means that the map from DNA codons to amino acids is many-one, leaving considerable freedom in how to genetically specify a particular protein sequence. The actual codons used depend on a number of factors including the transfer-RNA genes the organism has and various details of the biochemistry of the organism. This means that the fractions of use of each codon in a given gene is a good set of classification characters for that gene.

A large set of plant genes (1,636,317 sequences) was downloaded May 2015 from Phytozome 10.2 [9]. All of these plant species use the same standard genetic code for nuclear genes. DNA sequences are based on the protein-coding region of annotated, spliced mRNA transcripts with alternative splice variants, and include the stop codon. These plant genes were mined for their codon usage creating vectors of 64 positive values that sum to one. A set of 250 genes for the specific family of proteins called dehydrins that help plants deal with drought, cold, and high salinity [16] were included in this set of records. Dehydrin sequences were selected on the basis that gene.

Table I shows eighteen clusters that contained any of the 250 dehydrin records; 99 clusters contained no dehydrins. A \(\chi^2\) test for the null hypothesis that the dehydrin records are uniformly distributed across the bins that contain dehydrin records rejects the null hypothesis with \(p < 0.00001\).

Calling the binning of the codon-use records clustering is justified by the fact that the bin-defining points are well spaced out. The resulting clustering, however, will only be high-quality by accident. The advantage of performing a clustering with a point packing is that it is exceedingly fast, with algorithm time \(O(N)\) where \(N\) is the number of data records. Each data record is compared to a fixed number of points. In this case, this quick-and-dirty clustering method yielded a strong signal that dehydrins have an exceptional distribution of codon usage and so their codon usage merits additional investigation with more powerful and accurate tools. This exercise also highlights the usefulness of being able to pack points into a non-hypercuboidal domain.

When using point packings as initial populations for evolutionary optimizers, the hyperoctahedral modification of the point packings enabled much broader sampling of the search space. The constant sum one hyperplane is not invariant under face-swaps of the hypercube containing it and, in general, when the CVO is used to pack points into an oddly shaped set then there is the question of what symmetries that set has. For the constant sum hyperplane, the set we are using is the simplex that is the intersection of the hypercube \([0, 1]^{64}\) and the infinite plane of points with constant sum one. This set is a simplex with vertices equal to the standard basis in \(\mathbb{R}^{64}\) and so permuting the coordinates of the points preserves the simplex and we can still obtain \(63!\) alternate forms of the point packing used to bin codon use.

### VI. Conclusions and Next Steps

This study introduced the Conway variation operator for finding point packings in an arbitrary set. The unary and binary versions of the operator were examined in a parameter study and the binary version of the operator was found to be superior. The parameter study also established that large population size is helpful but that the random material rate, analogous to a mutation rate, has a different pattern of good settings on different cases of the point packing problem. Formal comparison with other methods of point packing remain unpublished at this time, with the exception of the 2-dimensional case. For packings into the unit square, decades of theorem-and-proof and careful algorithmic work provide a catalog of packings that we have, thus far, been unable to improve.

This study re-purposed the CVO from edit-metric spaces to Euclidean domains. Both of these are metric spaces in the classical sense and the CVO could be applied to finding well-spaced-out sets in any metric space. This means it could be used for feature selection relative to any metric on spaces of features, for example, or as an approach to the dominating set problem in graphs [20].

#### A. Other Representations

In preliminary work for this study, many other representations were cursorily examined. The direct and walking triangle representations in [2] had difficulty solving the problem for 8 or 9 points in the unit square while the CVO-based algorithm had no trouble working with much larger sets. A representation that evolves the vectors that are the basis for a lattice and then folds the lattice (mod 1) turned in results with 50-80%
of the fitness of the CVO-based method. This method had the advantage that only a small number of parameters was needed to specify a point packing, but the mutation operator for this algorithm moves points and so lacks the CVO-based algorithms ability to work in sets other than hypercubes. In contrast, a priority for future research is to examine a Baldwinian method, based on a CVO algorithm, since the inability of the CVO-based methods to move points means it is not using local optimization to improve the configurations it finds.

B. Applications

Three application domains were explored. It is difficult to judge the quality of the RGB color palettes, other than by examining them, but both the initialization of evolutionary optimizers and the fast clustering applications yielded good and interesting results. One flaw in the population initialization results was that the attempt to keep the population sizes the same involved a good deal of fiddling around with the minimum distance parameter \( \delta \) to get packings that were approximately the same size. A simpler way of dealing with this would have been to find packings of larger numbers of points and then sample those to obtain initial populations. A full point packing, however, covers the space fairly evenly meaning that random samples of a point packing might yield initialization performance intermediate between full point packings as initializers and random initialization. Re-running Conway’s lexicode algorithm with a larger exclusion distance than \( \delta \) might be a way to find a well-spaced subset of a packing.

C. Data Driven Point Packings

One strength of the CVO based evolutionary algorithm for finding point packings is that the set from which the points are drawn determines the set that points are being packed into. The biological application in this study was to use a point packing to cluster just over 1.6 million records. Because those records represent empirical distributions on a set of 64 possibilities, representing a well-characterized hyperplane, it was possible to generate random objects in that hyperplane to drive the CVO-based algorithm. A possible alternative approach is to use random selections of points from the data set to be clustered as the source of points for building packings. This approach will strengthen the ability of the CVO-algorithm to pack points into sets with a complex or unknown shape because it is drawing points directly from the set. In particular, when clustering, it is often the case that no set-theoretic or algorithmic description of the set from which the points to be clustered are drawn is available.

D. Spherical Targets

It is easy to generate random points on the surface of a sphere or hypersphere. Packings of points into a sphere, the Fejes Tth’s problem [8], has been around for a long time but still contains many open questions. The case of packing Fejes Tth’s problem [8], has been around for a long time. D. Spherical Targets

algorithmic description of the set from which the points to be clustered

packings with an expected density that is approximately the same size. A simpler way of dealing with this would have been to find packings of larger numbers of points and then sample those to obtain initial populations. A full point packing, however, covers the space fairly evenly meaning that random samples of a point packing might yield initialization performance intermediate between full point packings as initializers and random initialization. Re-running Conway’s lexicode algorithm with a larger exclusion distance than \( \delta \) might be a way to find a well-spaced subset of a packing.

C. Data Driven Point Packings

One strength of the CVO based evolutionary algorithm for finding point packings is that the set from which the points are drawn determines the set that points are being packed into. The biological application in this study was to use a point packing to cluster just over 1.6 million records. Because those records represent empirical distributions on a set of 64 possibilities, representing a well-characterized hyperplane, it was possible to generate random objects in that hyperplane to drive the CVO-based algorithm. A possible alternative approach is to use random selections of points from the data set to be clustered as the source of points for building packings. This approach will strengthen the ability of the CVO-algorithm to pack points into sets with a complex or unknown shape because it is drawing points directly from the set. In particular, when clustering, it is often the case that no set-theoretic or algorithmic description of the set from which the points to be clustered are drawn is available.

D. Spherical Targets

It is easy to generate random points on the surface of a sphere or hypersphere. Packings of points into a sphere, the Fejes Tth’s problem [8], has been around for a long time but still contains many open questions. The case of packing Fejes Tth’s problem [8], has been around for a long time. D. Spherical Targets

algorithmic description of the set from which the points to be clustered

packings with an expected density that is approximately the same size. A simpler way of dealing with this would have been to find packings of larger numbers of points and then sample those to obtain initial populations. A full point packing, however, covers the space fairly evenly meaning that random samples of a point packing might yield initialization performance intermediate between full point packings as initializers and random initialization. Re-running Conway’s lexicode algorithm with a larger exclusion distance than \( \delta \) might be a way to find a well-spaced subset of a packing.

C. Data Driven Point Packings

One strength of the CVO based evolutionary algorithm for finding point packings is that the set from which the points are drawn determines the set that points are being packed into. The biological application in this study was to use a point packing to cluster just over 1.6 million records. Because those records represent empirical distributions on a set of 64 possibilities, representing a well-characterized hyperplane, it was possible to generate random objects in that hyperplane to drive the CVO-based algorithm. A possible alternative approach is to use random selections of points from the data set to be clustered as the source of points for building packings. This approach will strengthen the ability of the CVO-algorithm to pack points into sets with a complex or unknown shape because it is drawing points directly from the set. In particular, when clustering, it is often the case that no set-theoretic or algorithmic description of the set from which the points to be clustered are drawn is available.

D. Spherical Targets

It is easy to generate random points on the surface of a sphere or hypersphere. Packings of points into a sphere, the Fejes Tth’s problem [8], has been around for a long time but still contains many open questions. The case of packing points onto a sphere is of substantial interest to cartographers

and geographers to enable geographically equitable samplings, though angular separation replaces Euclidean distance as the measure of interest. A great deal of information, for an upcoming book on spherical codes, is available from N. A. Sloan’s web site.

REFERENCES