An Efficient Implementation of Distance-Based Diversity Measures Based on *k*−*d* Trees

Dimitris K. Agrafiotis* and Victor S. Lobanov

3-Dimensional Pharmaceuticals, Inc., 665 Stockton Drive, Suite 104, Exton, Pennsylvania 19341

Received June 9, 1998

The problem of quantifying molecular diversity continues to attract significant interest among computational chemists. Most algorithms reported to date are distance-based and scale to the square of the size of the data set. This paper reports an alternative algorithm based on *k*-dimensional (or *k*−*d*) trees. *k*−*d* trees are combinatorial data structures that allow expedient location of nearest neighbors in multivariate spaces. Nearest neighbor detection forms the basis of many popular diversity measures, such as maximin, minimum spanning trees, and many others. In this report, we demonstrate that *k*−*d* trees exhibit excellent scaling characteristics and can be used to accelerate diversity estimation without compromising the quality of the design. The advantages of this approach are contrasted with an alternative algorithm that was recently proposed by Turner et al. based on the cosine similarity coefficient.

INTRODUCTION

In recent years, advances in synthetic and screening technology have enabled the simultaneous synthesis and biological evaluation of large chemical libraries containing hundreds to tens of thousands of compounds. Molecular diversity continues to be the main guiding principle in the design of combinatorial and high-throughput screening experiments and has become one of the most rapidly growing fields of computational chemistry. Since the publication of Martin's seminal paper on the use of statistical experimental design for reagent selection,[1] tens of articles have appeared in the literature dealing with the problem of diversity estimation and combinatorial library design.[2−4] From a computational perspective, the problem of quantifying diversity consists of two parts: the first is the definition of chemical space, and the second is the choice of a suitable measure. The former is notoriously hard and ill-defined since it is often context-dependent. The latter is conceptually simpler but suffers from the fact that validation is very difficult. A less controversial issue is that of performance. An ideal algorithm should be fast and scalable to allow the processing of the large data sets that are encountered in a typical library design application.

Most diversity measures reported to date are based on some measure of pairwise molecular similarity calculated using one or more molecular descriptors.[2−4] These algorithms tend to scale to the square of the number of compounds in the library, which imposes severe limitations in the size of the data sets that can be effectively analyzed using existing hardware. A number of solutions have been proposed to remedy this problem. For example, Willett's group has recently proposed a new diversity measure which, when combined with the cosine similarity coefficient, leads to an algorithm with linear time complexity.[5] As we demonstrate in this paper, the impressive performance of this algorithm is offset by a strong tendency to sample the extremes of the feature space and produce redundant designs.

Among the different strategies for maximizing diversity, perhaps the most commonly used is maximin, a recursive procedure that was originally proposed by Lajiness[6] and was later refined by Polinsky et al.[7] Maximin starts with a randomly chosen compound and gradually builds up the diversity list by examining the remaining compounds and selecting the one that is most different from the already selected members. Later, Agrafiotis[8,9] and Hassan et al.[10] independently proposed a selection algorithm in which maximin and some of its variants were cast as objective functions that were minimized by a Monte Carlo procedure. Unfortunately, both algorithms exhibit quadratic complexity which limits them in terms of scope and applicability.

This paper discusses a new efficient implementation of maximin and other distance-based diversity measures based on *k*-dimensional (or *k*−*d*) trees. *k*−*d* trees are combinatorial data structures that allow expedient location of nearest neighbors in multivariate spaces. Our goal is to demonstrate that it is possible to improve the performance of distance-based diversity measures algorithmically without compromising the quality of the design, which is a shortcoming of some other reported methodologies.

2. METHODS

2.1. *k*-Dimensional Trees. *k*-Dimensional or *k*−*d* trees are dynamic, adaptable data structures used to partition a geometric space in a manner convenient for use in range searching, nearest neighbor searching, and other related problems.[11,12] A *k*−*d* tree is a generalization of a regular binary tree used for sorting and searching. The tree partitions a set of multidimensional points, *S*, into smaller subsets using the coordinates of the points as discriminators. Each node in the tree represents a subset of *S* and a partitioning of that subset. The root of the tree represents the entire set *S*. Each nonterminal node partitions the points associated with that node according to their position with respect to a *k*-dimensional hyperplane; the points that lie to the "left" of that hyperplane are stored on the left subtree, while those that lie on the "right" of the hyperplane are stored on the

* Corresponding author: Tel: (610) 458-6045, Fax: (610) 458-8249, E-mail: dimitris@3dp.com.

10.1021/ci980100c CCC: $18.00 © 1999 American Chemical Society
Published on Web 11/17/1998

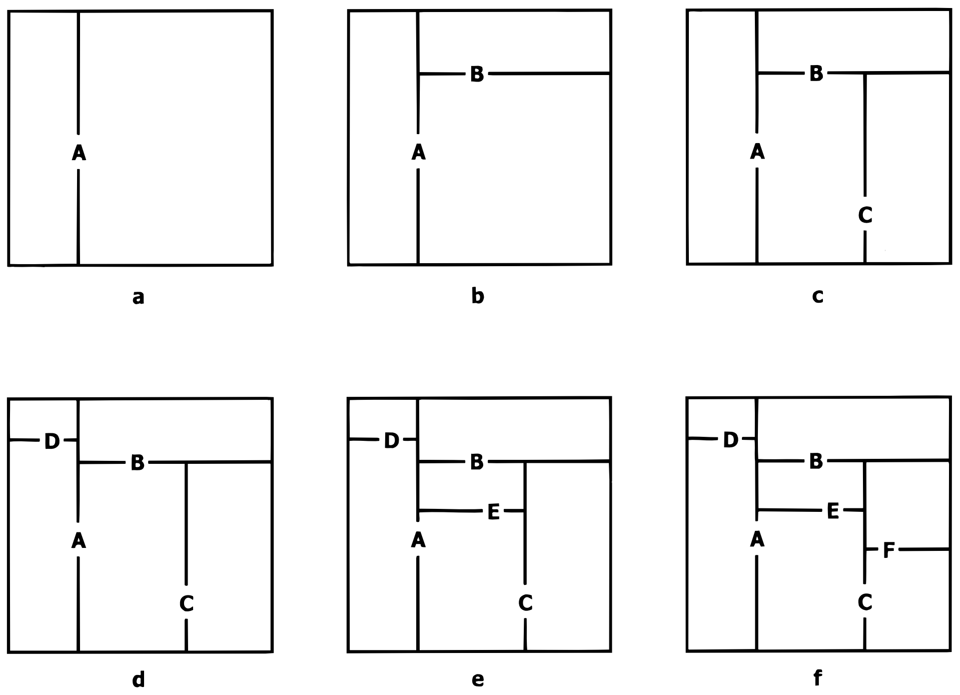

Figure 1. Partitioning of the plane with a 2-d tree. Points are inserted in the order A, B, C, D, E, and F.

right subtree. The tree can be elaborated to any depth, ending with terminal nodes which can hold up to a predetermined number of points. Thus, the terminal nodes define mutually exclusive small subsets of *S* (called bins), which collectively form a partition of *S*.

In its most straightforward implementation, the partition plane associated with a particular node is perpendicular to a coordinate axis which is chosen on the basis of the level of that node on the tree; that is, the discriminator is chosen by cycling through the coordinates in a strictly alternating sequence. Thus, all nodes on a given level of the tree have the same discriminator. If the coordinates are numbered from 1 to *k*, the root partitions the first axis, its two offspring partition the second axis, and so on to the *k*th level which partitions the *k*th axis. Once all the dimensions are exhausted, the partitioning begins again from the first axis; that is, the $k + 1$ level partitions the first axis, the $k + 2$ level partitions the second axis, etc.

The tree is constructed recursively in a manner similar to a regular binary tree. The process is illustrated in Figure 1 for a simple two-dimensional case. The points in the data set are processed in a sequential, random, or predetermined order. Each point is inserted by traversing the tree in a recursive manner starting from the root, each time following the branch determined by the specified coordinate of the current node and the respective coordinate of the new point. In the example in Figure 1, the points are inserted in the order A, B, C, D, E, and F. The process begins by selecting the first point, A, and inserting it at the root of the tree. Since the level of the root node is 1, A partitions the data space into two half-planes, as illustrated by the vertical line passing through A. Next, point B is examined. Since A partitions the *x* coordinate and B lies to the right of A, B is inserted to the right of A in the tree. B lies on the second level and partitions the right half-plane of A into two parts, as illustrated by the horizontal line passing through B. Point C is processed next. Since C lies to the right of A we go right

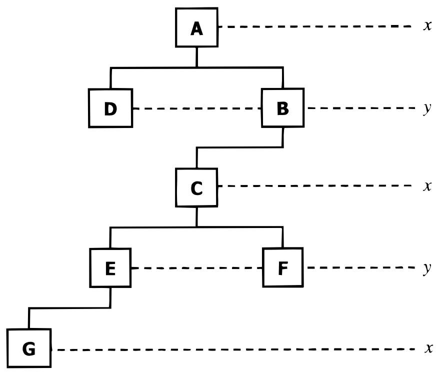

Figure 2. 2-d tree resulting from the partitioning of the points in Figure 1.

at the root, and since it is below B we insert C to the left of B. The insertion of D proceeds along the same lines, going left at the root since D lies to the left of A. E is processed next, followed by F and G. The partitioning of the space after the insertion of all six points is shown in Figure 1f and the resulting tree in Figure 2.

Every terminal node of the $k-d$ tree corresponds to some rectangle in the plane, and this feature makes them ideally suited for range and nearest-neighbor searching. The nearest-neighbor search algorithm takes as input a query point, Q, and descends the nonterminal nodes of the $k-d$ tree, choosing at each node to investigate either the left or right child according to whether the query lies to the “left” or “right” side of the partition plane. When the search reaches a terminal node, the points in that node are examined exhaustively and the point nearest to the query is identified. This point C represents the current nearest neighbor of Q, and its distance from Q is denoted as d_C. Unfortunately, as illustrated in Figure 3, the search is not complete at this point. Figure 3a shows a query Q, its current nearest neighbor C, and a hypothetical vertical partition plane in a two-dimensional space. Both the query, Q, and the current nearest neighbor,

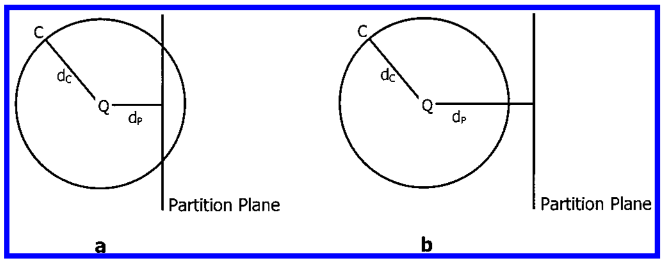

Figure 3. Bounds-overlap-ball test. Q denotes the query, C its current nearest neighbor, d_c the distance of the query from C, and d_p the distance of the query from the partition plane; (a) the partition plane lies close to Q than C, and thus the other side must be searched; (b) C lies closer to Q than the partition plane, and thus the other side need not be searched.

C, lie to the left of that plane. If there is a point closer to Q than its current nearest neighbor C, it must lie within a circle of radius d_C centered at Q. Since part of this circle lies to the right of the partition plane, the algorithm must search the points on the other side of the plane to determine whether the nearest neighbor has been found. However, if the current nearest neighbor lies closer to the query than the partition plane, a situation illustrated in Figure 3b, it is not necessary to test the other side of the plane. Indeed, to decide whether the other side of the plane must be searched, the algorithm need only compare the distances of the query from the current nearest neighbor and the partition plane, respectively, and this is true regardless of the dimensionality of the space.

Thus, when the search reaches a nonterminal node, the recursive procedure is called for the node that represents the subtree on the same side of the partition plane as the query. When control returns, a test is made to determine if it is necessary to consider the opposite side of the plane by comparing the distances of the query from the current nearest neighbor and the partition plane. Friedman, Bentley, and Finkel who introduced the original algorithm refer to this as the “bounds-overlap-ball” test.[12] If the test fails, the records on the opposite side need not be considered, and the procedure backtracks. If not, the procedure is called recursively for the node representing that subtree. In both cases, before backtracking to the parent node, a second test is made to determine if it is necessary to continue the search. This criterion, referred to as the “ball-within-bounds” test, determines whether the ball lies entirely within the geometric domain of the current node, in which case the search is terminated. The algorithm is described in greater detail in ref 12.

Friedman et al. showed that both insertion and search scale logarithmically with the size of the data set.[12] In particular, the time required to organize a k-dimensional tree of N random data points is asymptotically proportional to N log N, while the time required to search for a nearest neighbor is proportional to log N. These attractive scaling properties make $k-d$ trees ideally suited for nearest-neighbor searching and form the basis of our implementation of maximin and other related diversity measures described below.

The $k-d$ tree algorithm described above is quite general and does not require that the partition planes are perpendicular to the coordinate axes. For example, Sproull[13] proposed an alternative algorithm in which the partition plane can have an arbitrary orientation determined by the principal eigenvector of the covariance matrix of the points to be partitioned. Sproull’s algorithm is more effective with highly skewed and unevenly distributed data. However, there is a significant overhead associated with principal component analysis, which makes it impractical for the application at hand. We should also point out that the search algorithm is only asymptotically logarithmic in N, and it is wrong to assume that nearest neighbors *will* be found in logarithmic time, particularly for high-dimensional spaces and small, uniformly distributed data sets. A number of approximate algorithms have been proposed that find nearby points without being the nearest, that do operate in logarithmic time. In addition, a number of promising alternative data structures have been proposed, such as bump-trees,[14] ball-trees,[15] oct-trees,[16] and vantage-point trees[17] to name a few, and their potential use in diversity analysis is currently under investigation.

2.2. Diversity Measure. The method described herein is based on the algorithm we introduced in refs 8 and 9. In particular, each compound is represented as a k-dimensional vector of independent continuous variables, computed either directly in the form of molecular descriptors or indirectly through principal component analysis, multidimensional scaling, factor analysis, or some other equivalent technique. The diversity D(S) of a given set of compounds, S, is given by the maximin function

$$\mathrm{D(S)} = \max_{i}\left(\min_{j\neq i}(d_{\mathrm{ij}})\right) \tag{1}$$

where i, j are used to index the elements of S, and d_{ij} is the Euclidean distance between the ith and jth compounds. Upon closer inspection, it can be seen that eq 1 is relatively “discontinuous” and can be easily dominated by the presence of an outlier. It is often desirable to have a “smoother” diversity function that can discriminate more effectively between the various ensembles. In the algorithm that we originally proposed (vide infra), diversity is cast as an objective function that is optimized using a Monte Carlo procedure such as simulated annealing or evolutionary programming. In this context, we have found the mean nearest neighbor distance (eq 2) to be a more appropriate choice

$$\mathrm{D(S)} = \frac{1}{m}\sum_{i}\left(\min_{j\neq i}(d_{\mathrm{ij}})\right) \tag{2}$$

where m is the number of compounds in S.

Naively implemented, eqs 1 and 2 require $m(m-1)/2$ distance computations, i.e., they scale to the square of the number of compounds in the data set and can be prohibitively expensive for large collections. However, the complexity of the problem stems from the need to determine the distance of each point from its nearest neighbor in the data set, which, as we pointed out above, can be dramatically improved using an efficient data structure such as a k-dimensional tree. The approach taken here is to organize all the compounds into a $k-d$ tree and then perform a nearest neighbor search for each compound in the data set. Thus, the algorithm involves two passes through the data set: a first pass where the tree is constructed, and a second pass where every compound is used as a query in a nearest neighbor search. Since, according to Friedman, both phases are of O(N log N), the whole process exhibits N log N time complexity. The examples described below demonstrate that this is indeed the case and prove the usefulness of this approach in diversity profiling and combinatorial library design.

2.3. Compound Selection. The performance of the method described above was tested in the context of a Monte Carlo algorithm for compound selection and combinatorial library design. The algorithm is described in detail in refs 8 and 9 and is summarized below for the sake of completeness. The problem can be stated as follows: given an n-membered virtual library and a number m, find the m most diverse compounds in that population. Our solution to this problem was to cast diversity in the form of a fitness function and use a stochastic search engine such as simulated annealing or evolutionary programming to identify the optimal (most diverse) set.[8,9] Simulated annealing starts from an initial random state and walks through the state space associated with the problem of interest by a series of small stochastic steps. An objective function maps each state into a real value, which measures its energy or fitness. In the problem at hand, a state is a unique m-membered subset of the n-membered virtual library, its fitness is the diversity of that set as measured by eq 2, and the step is a small change in the composition of that set (usually some small fraction of the points comprising the set). While downhill transitions are always accepted, uphill transitions are accepted with a probability that is inversely proportional to the energy difference between the two states. This probability is controlled by a parameter called temperature, which is adjusted in a systematic manner during the course of the simulation.

The power of this method stems from its generality, i.e., the fact that the selection can be carried out based on any desirable selection criterion. Examples of such alternative criteria including similarity to existing leads, log P profile, heteroatom content, and ease of synthesis are given on an accompanying paper.[18] Here, the selection criterion was to maximize the diversity of the design as measured by eq 2, subject to the constraint that the number of compounds in the final solution should be exactly m. The simulations were carried out in 30 temperature cycles, using 1000 sampling steps per cycle, a Gaussian cooling schedule, and the Metropolis acceptance criterion ($p = e^{-\Delta E/K_B T}$). Boltzmann's "constant", K_B, was adjusted in an adaptive manner, by constantly updating the mean transition energy during the course of the simulation and continuously adjusting the value of K_B so that the acceptance probability for a mean uphill transition at the final temperature was 0.1%.

The algorithm described above involves the evaluation of eq 2 for every state in the annealing chain. This is carried out by first organizing all the points comprising the current state into a $k-d$ tree and then performing m nearest neighbor searches, one for each point in the state. In our implementation, the trees are elaborated to maximum depth, i.e., each terminal node contains a single compound. To avoid any order dependencies and to ensure that the resulting trees are balanced, the points are processed in random order.

2.4. Sofware. The software was implemented using an object-oriented approach and the C++ programming language. It is based on 3-Dimensional Pharmaceuticals' generic C++ simulation classes and the Mt++ toolkit.[19] The calculations were performed on an R10K Silicon Graphics Challenge workstation running Irix 6.2.

Table 1. Timings for 1000 Function Evaluations using the Naïve Algorithm

	2D	3D	4D	5D	6D	7D	8D
10	0.4	0.4	0.4	0.4	0.4	0.4	0.5
20	1.4	1.4	1.6	1.9	1.9	1.9	2.0
40	5.6	5.8	6.6	7.5	7.6	7.6	7.9
80	22.3	23.2	26.5	29.8	30.4	30.9	31.4
160	89.4	92.5	105.4	118.5	119.0	123.3	126.0
320	358.2	369.1	421.0	472.0	472.8	492.0	502.0
640	1434.2	1474.0	1682.1	1885.5	1889.2	1966.2	2007.1
scaling	4.00	3.99	3.99	3.99	4.00	4.00	4.00

Table 2. Timings for 1000 Function Evaluations using the $k-d$ Tree Algorithm

	2D	3D	4D	5D	6D	7D	8D
10	0.3	0.3	0.3	0.3	0.4	0.4	0.4
20	0.6	0.7	0.8	0.9	1.0	1.1	1.1
40	1.3	1.7	2.1	2.6	3.0	3.4	3.7
80	2.8	4.0	5.4	6.8	8.6	10.5	11.8
160	6.1	9.2	13.3	18.3	24.5	30.9	38.1
320	13.0	20.7	31.5	46.3	64.9	87.9	115.7
640	28.0	45.9	74.9	111.7	169.6	246.2	333.9
scaling	2.15	2.22	2.38	2.41	2.61	2.8	2.89

3. RESULTS AND DISCUSSION

3.1. Performance of k-Dimensional Trees. To assess the performance of $k-d$ trees, the algorithm was tested on a series of two to eight-dimensional artificial data sets comprised of 10 000 points uniformly distributed in the unit hypercube. To minimize data set dependencies and order bias, we measured the time required to perform 1000 fitness evaluations on k-membered random subsets of the data sets, where k ranges from 10 to 640. Moreover, to mimic the annealing environment described in section 2.3, each fitness evaluation was followed by a random single point mutation of the k-membered set. For all but the smallest selections (i.e., where k is small), the calculation is dominated by the fitness evaluation and thus reflects the true computational demands of the $k-d$ tree algorithm.

Tables 1 and 2 show the timings obtained on an R10K SGI Challenge workstation for 1000 function evaluations of 10−640-membered subsets in two to eight dimensions with (Table 2) and without (Table 1) the use of $k-d$ trees. The last row in the tables (labeled "scaling") represents the scaling of the algorithm for each dimensionality, computed as the ratio of T_{640}/T_{320}, where T_k is the time required to compute 1000 fitness evaluations of k points. This value represents the relative increase in CPU time observed when the number of points in the selection doubles. Although this factor varies slightly depending on which pair is examined (e.g. T_{640}/T_{320} vs T_{320}/T_{160}), the higher numbers are closer to the "true" value since a relatively greater fraction of the CPU time is spent evaluating the fitness function. This value should be 2 for a perfectly linear algorithm and 4 for a perfectly quadratic one.

The effect of $k-d$ trees in two dimensions is shown graphically in Figure 4. It is clear that while for very low values of k $k-d$ trees impose some trivial overhead, they scale superbly with k, resulting in enormous computational savings for large selections. For low dimensions, the scaling is less than 2.5 which is consistent with the reported theoretical value of $N \log N$. For example, for 2D data the algorithm scales essentially linearly with k, since doubling the number of points only causes a 2.15 increase in the CPU requirements of the algorithm.

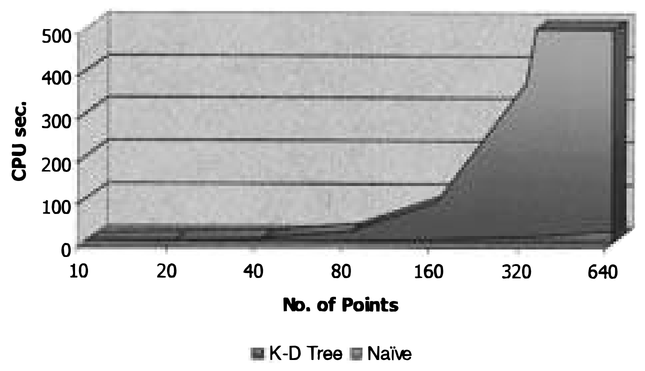

Figure 4. Scaling of the naïve and $k-d$ tree algorithms as a function of size (2D).

However, our studies also indicate that this margin is decreased in higher dimensions. For example, the scaling factor is 2.22 for 3D, 2.38 for 4D, and 2.89 for 8D, a trend shown graphically in Figure 5. Thus, for eight dimensions, the time complexity is somewhere between linear and quadratic. Although we did not carry out simulations beyond eight dimensions, this performance decline is likely to continue, and the algorithm will eventually become quadratic, with the added overhead of constructing and traversing the tree. However, our experience has shown that most high-dimensional descriptor spaces can be reduced to three to eight dimensions with minimal loss of information through the use of efficient nonlinear mapping techniques.[20]

3.2. Comparative Results. To demonstrate the advantages of $k-d$ trees, we decided to compare it with an alternative algorithm designed specifically to deal with the quadratic complexity of the diversity problem. In particular, this section discusses a method that was recently proposed by Willett and co-workers based on the cosine coefficient of similarity.[5] Our analysis follows the spirit of our recent study[21] of Lin's[22] information-theoretic approach for assessing molecular diversity. Lin's approach was based on the idea that the diversity of a given set of compounds could be directly related to its information content, as quantified by Shannon's classical entropy equation. While the idea was novel and interesting, two misconceptions associated with the sign of entropy and the implicit requirement that all compounds be equidistant caused it to collapse even in the most trivial one-dimensional cases. The analysis presented here follows closely that example and leads to similar conclusions.

Willett proposed that the diversity of a set of compounds, A, be defined by the mean pairwise intermolecular dissimilarity

$$\mathrm{D(S)} = 1 - \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\sigma(i,j)}{N^2} \tag{3}$$

where $\sigma(i, j)$ is the similarity between two compounds, i and j, in S, and N is the total number of compounds in S. With appropriate normalization, the diversity index, D(S), can be confined in the interval [0, 1] (i.e., $0 \le \mathrm{D(S)} \le 1$). A value of 1 indicates a completely diverse collection, i.e., one where all compounds have zero similarity to each other, while a value of 0 indicates a collection of identical structures (or identical descriptions, to be more precise).

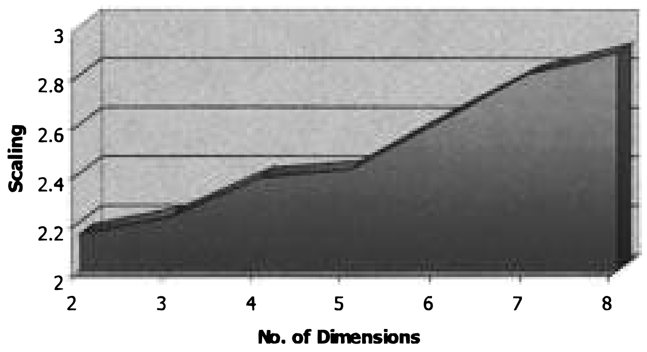

Figure 5. Scaling of the $k-d$ tree algorithm as a function of dimensionality.

Like many indices of this kind (including eqs 1 and 2), eq 3 exhibits quadratic complexity. That is, to evaluate D(S) one needs to compute the similarity matrix $\sigma(i, j)$, which scales to the square of the number of compounds in the data set. However, if the cosine coefficient is employed to compute the pairwise similarities, eq 3 can be transformed into a functional form that can be evaluated in linear time. The cosine coefficient is applicable to any situation in which the compounds can be represented in a vectorial form, e.g. by a set of topological indices or computed molecular properties. It has been used extensively in information retrieval systems, and is defined as the cosine of the angle formed by two molecular property vectors

$$\cos(i, j) = \frac{\sum_{k=1}^{K} m(i, k)m(j, k)}{\sqrt{\sum_{k=1}^{K} m(i, k)^2 \sum_{k=1}^{K} m(j, k)^2}} \tag{4}$$

where $m(i, k)$ represents the kth feature of the ith compound in the data set. By virtue of eq 4, the numerator in eq 3 becomes

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\cos(i, j) = \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\sum_{k=1}^{K} m(i, k)m(j, k)}{\sqrt{\sum_{k=1}^{K} m(i, k)^2}\sqrt{\sum_{k=1}^{K} m(j, k)^2}} \tag{5}$$

If we define $w(i)$ to be the "weight" of the ith compound in S

$$w(i) = \frac{1}{\sqrt{\sum_{k=1}^{K} m(i, k)}} \tag{6}$$

and α_c to be the linear combination or "centroid" of the individual molecular vectors, $m(i)$

$$a_c(j) = \sum_{i=1}^{N} w(i)m(i, j) \tag{7}$$

Equation 5 can be rewritten as

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\cos(i,j) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} w(i)w(j)m(i,k)m(j,k) = (\sum_{i=1}^{N} w(i)m(i))(\sum_{j=1}^{N} w(j)m(j)) = \mathbf{a}_c\cdot\mathbf{a}_c \quad (8)$$

and eq 3 becomes

$$\mathrm{D(A)} = 1 - \frac{\mathbf{a}_c\cdot\mathbf{a}_c}{N^2} \quad (9)$$

where $\mathbf{a}_c\cdot\mathbf{a}_c$ is the dot product of the vector a_c with itself. Since the evaluation of a_c requires a single pass through the data set, this algorithm is of linear order.

Although this algorithm executes at a blazing speed that permits the processing of very large data sets, this performance improvement comes at a high price. Consider the following example.

Figure 6 shows two different sets of three imaginary compounds plotted against a uniform property scale. Since the cosine coefficient is only concerned with the angles between the property vectors and not their lengths, according to eq 9, these two sets should appear equally diverse (the angle ∂ is the same in both data sets). When applied on a larger scale, this assumption can have profound effects on the quality of the design. The cosine coefficient does not maximize spread, only orthogonality. Whether this is sufficient will become more evident in the next few paragraphs. At this point, it is sufficient to note that while orthogonality is important in structure−activity correlation, orthogonality criteria *alone* cannot distinguish the extremes from the intermediate regions of the feature space, which are inherently more interesting in drug-related applications.

Of course, the similarity index is only one of the two main components that make up a diversity index. The other is the *diversity measure*, i.e., the function that measures the diversity of a data set as a function (usually) of the pairwise similarities of its members. In fact, the choice of the diversity measure is often more critical than that of the similarity index. Consider again the example shown in Figure 6a. Figure 7 illustrates the diversity of this data set as a function of the position of the third point with respect to the other two. The plot was generated by varying the angle ∂ between points 1 and 2 while keeping 1 and 3 fixed and computing the diversity of the set at each position using eq 9.

Interestingly, the function increases monotonically as the third point moves away from the diagonal and reaches a maximum when it coincides with one of the two reference points. This is a very troubling result, the reasons for which can be easily understood by simple trigonometry. Since points 1 and 3 are held fixed, the variable part of eq 3 is the sum of the cosines between vectors 1−2 and 2−3, s(∂). By applying the well-known trigonometric relationship

$$\cos a + \cos b = 2\cos\frac{a+b}{2}\cos\frac{a-b}{2} \quad (10)$$

this sum is given by

$$s(\vartheta) = \cos(\vartheta) + \cos(90-\vartheta) = 2\cos\left(\frac{90}{2}\right)\cos\left(\frac{90-2\vartheta}{2}\right) = 2\cos(45)\cos(45-\vartheta) \quad (11)$$

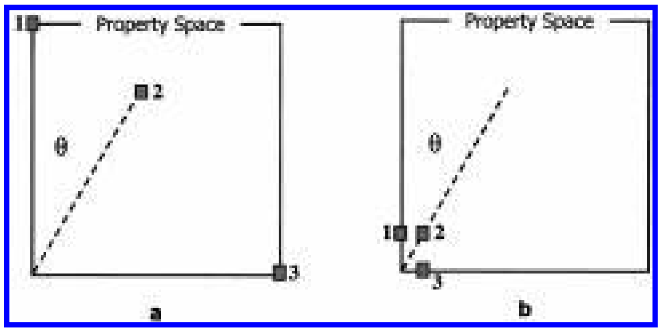

Figure 6. Two simple data sets, each comprised of three points distributed in the unit square.

Thus

$$\max_{\vartheta}(s(\vartheta)) = \max_{\vartheta}\cos(45-\vartheta) \quad (12)$$

and since $\partial \in [0,90]$, it follows that s(∂) is maximized (and D(S) is minimized) at $\partial = 45$. In fact, it is straightforward to show that this is true for any number of points confined within the unit square.

To get a better understanding of the properties of this measure, we applied it to the problem of compound selection described in section 2.3. Our analysis was based on two artificial data sets comprised of 10 000 points uniformly distributed in the unit square and cube, respectively. The property space is densely populated, does not exhibit any significant clustering, and is designed to reveal the structure of the “true” minimum. The selections were limited to 100 compounds and were performed using the procedure and simulation parameters described in section 2.3. The results are shown in Figures 8 and 9, respectively. In both cases, the selected points are distributed in *d* equally populated clusters located along the principal axes of the feature space, where *d* is the number of dimensions. This is consistent with our trigonometric analysis and is in fact true for any number *k* and dimensionality *d*.

An approximate numerical estimate of how well the selection samples the property space can be obtained from an eigen-analysis of the variance-covariance matrix of the selected compounds. In the 2D case, principal component analysis revealed two latent variables which accounted for 90.3% and 9.7% of the total variance respectively, indicating a very poor sampling of the feature space. Similarly, in the 3D case, the three PCs accounted for 50.7%, 44.4%, and 4.9% of the variance, respectively, which shows that one dimension is virtually lost in the design. These results are to be contrasted to maximin designs, which showed a uniform sampling of the property space with equally important PCs (51.1% and 48.9% for the 2D data and 34.5%, 33.2%, and 33.3% in the 3D case). We should also point out that the fact that there seems to be some variance along the principal axes is purely accidental and is not at all guaranteed by the algorithm. In fact, there is nothing in the measure itself that force points away from each other, which is what prohibits the formation of clusters and avoids oversampling.

On the basis of the results above, one would be tempted to think that the reason for these extreme designs is the cosine coefficient. This is not entirely true. The main reason is in fact the simplistic summation function in eq 3. This diversity

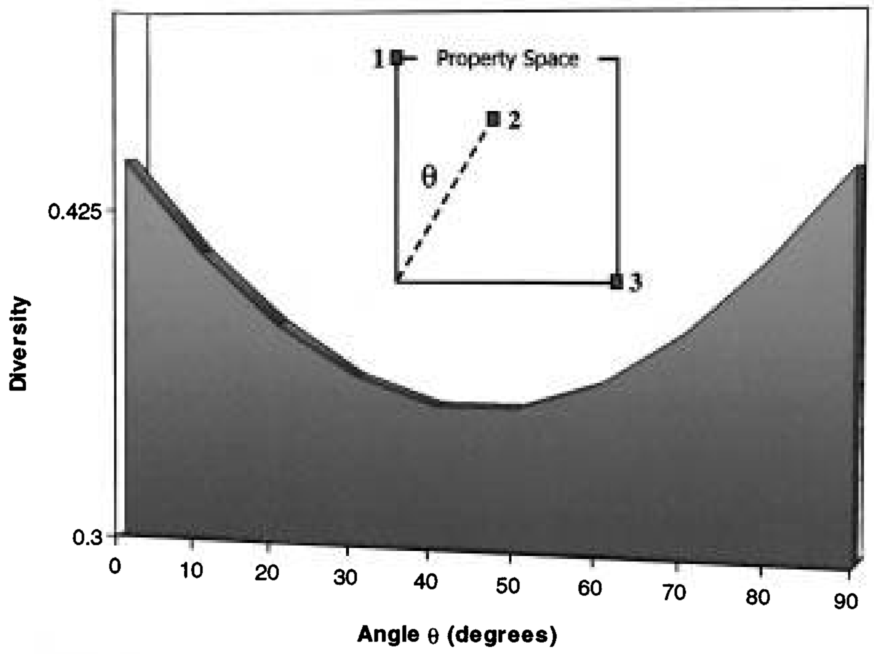

Figure 7. Turner's diversity of a three-point data set as a function of the position of the third point with respect to the other two.

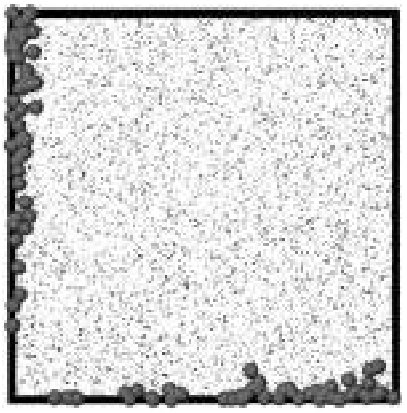

Figure 8. Selection of 100 points from a 2D uniform data set using Turner's diversity measure (dots: uniform data set; spheres: selected points).

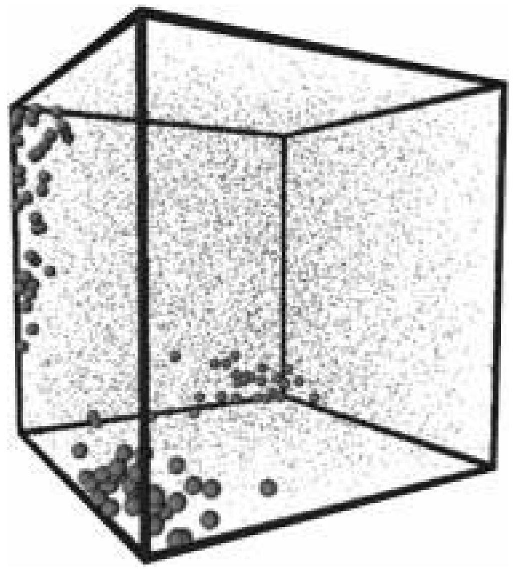

Figure 9. Selection of 100 points from a 3D uniform data set using Turner's diversity measure (dots: uniform data set; spheres: selected points).

measure would render extreme designs *regardless* of the similarity measure. To prove this, we substituted the cosine coefficient with the Euclidean norm and performed a selection of 25 points. The results are shown in Figure 10.

While the appearance of the design changes, its extremity remains the same. By maximizing the average pairwise distance, the measure favors the creation of dense clusters located at maximum separation.

Although the preceding analysis may appear abstract, it has significant implications in the study molecular diversity. It is known to any practicing chemist, experimental and computational alike, that the extremes of the chemical space represent peculiar structures of little intrinsic interest. Depending on the molecular representation, these regions represent extreme topologies (highly branched systems, polycyclics, polyhalogeneated compounds, long aliphatic chains, etc.), extreme physicochemical properties (high or low solubility or molecular weight), extreme three-dimensional structure, or all of the above. Since molecular diversity is a design strategy aimed at maximizing the hit-rate of high-throughput screening experiments, we need a design mechanism that avoids these extreme regions and forces the selection into the interior of the property space.

The behavior outlined above was also noticed by Willett and his group[23] shortly after they published their original algorithm. Snarey et al. compared several maximum dissimilarity and sphere-exclusion algorithms for dissimilarity selection, using the World Drug Index as a refrence set. They concluded that maxsum exhibits a bias toward compounds located near the edge of the data set; compounds located near the center of the feature space are selected only when all of the outlying regions have been thoroughly sampled. This resulted in subsets exhibiting a lower number of different activity classes in the WDI than did the maxmin and SE-minmax algorithms (see ref 23 for details). Whether this bias is in general desirable may be argued, but it is clearly not what one would expect from a "diverse" design.

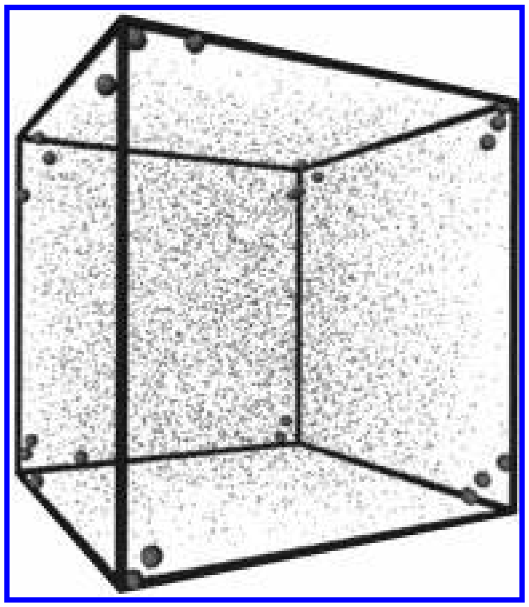

Figure 10. Selection of 25 points from a 3D uniform data set using eq 3 and the Euclidean metric as a similarity index (dots: uniform data set; spheres: selected points).

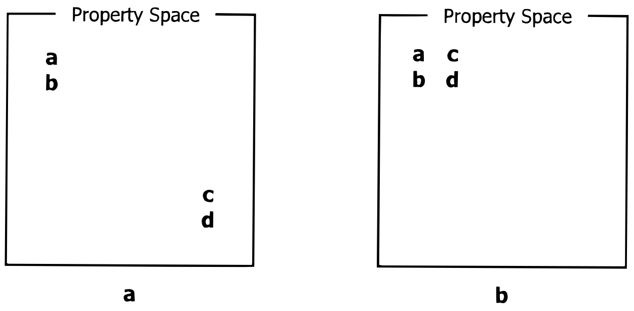

Figure 11. Limitations of maximin and related measures.

Finally, we should point out that nearest-neighbor-based diversity measures such as eqs 1 and 2 are themselves not flawless. A well-known deficiency of such measures is the inability to discriminate between the situations depicted in Figure 11. Indeed, the relative separation between the two clusters a,b and c,d is not taken into consideration since the measures are only concerned with the intracluster (nearest neighbor) separations. In practice, however, these defects are “annealed away” during the optimization process described in section 2.3. An interesting alternative was proposed by Mount et al.[24] and later by Waldman[25] and involves the use of minimum spanning trees. While an analysis of minimum spanning trees would not be appropriate in this report since the original work has yet to appear in print, we would like to point out that the construction of a minimum spanning tree also involves nearest neighbor detection as the most intensive step and could benefit greatly from the use of the $k-d$ tree algorithm described herein.

4. CONCLUSIONS

This paper describes an efficient implementation of nearest-neighbor-based diversity measures based on k-dimensional trees. The algorithm alleviates the quadratic time complexity of the diversity problem without compromising the quality of the design, which is a typical shortcoming of some other proposed methodologies. The algorithm is general and can be used in many different applications that involve nearest neighbor detection, including many diversity measures, nearest-neighbor classifiers and many others.

ACKNOWLEDGMENT

The authors would like to thank Dr. F. Raymond Salemme of 3-Dimensional Pharmaceuticals, Inc. for his insightful comments and support of this work.

REFERENCES AND NOTES

(1) Martin, E. J.; Blaney, J. M.; Siani, M. A.; Spellmeyer, D. C.; Wong, A. K.; Moos, W. H. *J. Med. Chem.* **1995**, *38*, 1431−1436.
(2) Agrafiotis, D. K. Molecular Diversity. In *Encyclopedia of Computational Chemistry*; Wiley: New York, in press.
(3) Agrafiotis, D. K.; Myslik, J. M.; Salemme, F. R. Advances in diversity profiling and combinatorial series design. In *Annual Reviews in Combinatorial Chemistry and Molecular Diversity*; in press.
(4) Martin, E. J.; Spellmeyer, D. C.; Critchlow, R. E., Jr.; Blaney, J. M. Does combinatorial chemistry obviate computer-aided drug design? In *Reviews in Computational Chemistry*; Lipkowitz, K. B.; Boyd, D. B., Eds.; VCH: New York, 1997; Vol. 10, Chapter 2, p 75.
(5) Turner, D. B.; Tyrrell, S. M.; Willett, P. Rapid quantification of molecular diversity for selective database aqcuisition. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 18−22.
(6) Lajiness, M. S. In *QSAR: Rational Aproaches to the Design of Bioactive Compounds*; Silipo, C., Vittoria, A., Eds.; Elsevier: Amsterdam, 1991; pp 201−204.
(7) Polinsky, A.; Feinstein, R. D.; Shi, S.; Kuki, A. In *Molecular Diversity and Combinatorial Chemistry*; Chaiken, I. M.; Janda, K. D., Eds.; American Chemical Society: Washington DC, 1996; pp 219−232.
(8) Agrafiotis, D. K. Stochastic algorithms for maximizing molecular diversity. 3-rd Electronic Computational Chemistry Conference, http://hackberry.chem.niu.edu/ECCC3/paper48, 1996.
(9) Agrafiotis, D. K. Stochastic algorithms for maximizing molecular diversity. *J. Chem. Inf. Comput. Sci.* **1997**, *37(5)*, 841−851.
(10) Hassan, M.; Bielawski, J. P.; Hempel, J. C.; Waldman, M. *Mol. Divers.* **1996**, *2*, 64−74.
(11) Bentley, J. L. Multidimensional binary search trees used for associative searching. *Comm. ACM* **1975**, *18(9)*, 509−517.
(12) Friedman, J. H.; Bentley, J. L.; Finkel, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Soft.* **1977**, *3(3)*, 209−226.
(13) Sproull, R. F. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica* **1991**, *6*, 579−589.
(14) Omohundro, S. M. Bump trees for efficient function, constraint and classification learning. In *Advances in Neural Information Processing Systems 3*; Lippmann, Moody, Touretzky, Eds.; Morgan-Kaufmann: San Mateo, CA, 1991; pp 693−699.
(15) Omohundro, S. M. Five balltree construction algorithms. *International Computer Science Institute Technical Report TR-89-063*, 1989, Berkeley, CA.
(16) Samet, H. The quad-tree and related hierarchical data structures. *ACM Comput. Surveys* **1984**, *16(2)*, 187−260.
(17) Yanilos, P. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. Fourth ACM-SIAM Sympos. Discrete Algorithms* **1993**, 311−321.
(18) Agrafiotis, D. K.; Lobanov, V. S.; Rassokhin, D. R., paper in preparation.
(19) Copyright 3-Dimensional Pharmaceuticals, Inc., 1994−1998.
(20) Agrafiotis, D. K.; Lobanov, V. S, Salemme, F. R. patents pending.
(21) Agrafiotis, D. K. On the use of information theory for assessing molecular diversity. *J. Chem. Inf. Comput. Sci.* **1997**, *37*(3), 576−580.
(22) Lin, S. K. Molecular diversity assessment: logarithmic relations of information and species diversity and logarithmic relations of entropy and indistinguishability after rejection of Gibbs paradox of entropy mixing. *Molecules* **1996**, *1*, 57−67.
(23) Snarey, M.; Terrett, N. K.; Willett, P.; Wilton, D. J. Comparison of algorithms for dissimilarity based compound selection. *J. Mol. Graphics Mod.*, in press.
(24) Mount, J.; Ruppert, J.; Welch, W.; Jain, A., *IBC 6-th Annual Conference on Rational Drug Design*; December 11−12, 1996; Coronado, CA.
(25) Waldman, M. *Symposium on Diverse Perspectives of Molecular Diversity*; ACS National Meeting, Dallas, TX, 1998.

CI980100C