Scalable Partitioning and Exploration of Chemical Spaces Using Geometric Hashing

Debojyoti Dutta,[†,§] Rajarshi Guha,[‡,§] Peter C. Jurs,*[,‡] and Ting Chen[†]

Department of Computational Biology, University of Southern California, Los Angeles, California 90089, and Department of Chemistry, Pennsylvania State University, University Park, Pennsylvania 16802

Received September 14, 2005

Virtual screening (VS) has become a preferred tool to augment high-throughput screening[1] and determine new leads in the drug discovery process. The core of a VS informatics pipeline includes several data mining algorithms that work on huge databases of chemical compounds containing millions of molecular structures and their associated data. Thus, scaling traditional applications such as classification, partitioning, and outlier detection for huge chemical data sets without a significant loss in accuracy is very important. In this paper, we introduce a data mining framework built on top of a recently developed fast approximate nearest-neighbor-finding algorithm[2] called locality-sensitive hashing (LSH) that can be used to mine huge chemical spaces in a scalable fashion using very modest computational resources. The core LSH algorithm hashes chemical descriptors so that points close to each other in the descriptor space are also close to each other in the hashed space. Using this data structure, one can perform approximate nearest-neighbor searches very quickly, in sublinear time. We validate the accuracy and performance of our framework on three real data sets of sizes ranging from 4337 to 249 071 molecules. Results indicate that the identification of nearest neighbors using the LSH algorithm is at least 2 orders of magnitude faster than the traditional *k*-nearest-neighbor method and is over 94% accurate for most query parameters. Furthermore, when viewed as a data-partitioning procedure, the LSH algorithm lends itself to easy parallelization of nearest-neighbor classification or regression. We also apply our framework to detect outlying (diverse) compounds in a given chemical space; this algorithm is extremely rapid in determining whether a compound is located in a sparse region of chemical space or not, and it is quite accurate when compared to results obtained using principal-component-analysis-based heuristics.

1. INTRODUCTION

In the past two decades, virtual screening (VS) has become a very popular method to speed up the drug discovery process, and it promises to effectively complement high-throughput screening.[1,3] One of the fundamental goals of VS is to find interesting starting points for further investigation. In VS, one typically handles huge virtual chemical libraries containing millions of small molecules. To determine targets early, such libraries need to be mined for a variety of tasks. For example, we may want to classify new molecules into classes such as inactive or active. Another interesting problem is to quickly identify the sparse regions of the chemical descriptor space for new leads. Mining such huge virtual libraries efficiently and accurately is an important and challenging problem.

One of the most common problems in mining large chemical libraries is *classifying* the compound into different *classes*. Different classes could represent different levels of activity or could represent different types of compounds. It is common to have the class labels represent disjointed sets of compounds of a given database. These classification techniques are often based on algorithms to *partition* the data set and then correctly assign a class label to each of the members of the cluster or partition. Even though clustering and classification are considered to be separate problems,[4] they are fundamentally related. Clustering is a useful preprocessing step before the actual classification. Also, classification can be considered to be a clustering problem, for example, if we want to partition the data set into *k* unique classes of activities. Thus, we would like to have a method that can be used for classification and clustering at the same time. Both these tasks can be solved using the solutions to a *k*-nearest-neighbor (*k*NN) problem. For example, it is well-known that the asymptotic error of a nearest-neighbor classifier is at most twice that of Bayesian classification.[5]

Another important data mining problem is finding the sparse regions in a chemical space defined by a set of molecular descriptors. This is a member of a class of problems called chemical diversity analysis.[6−8] A chemical descriptor space is a multidimensional space described by a vector of chemical descriptors. Unless mentioned otherwise, we assume that descriptors are real-valued. There are various other kinds of descriptors as well,[9] and our techniques may also apply to them with some extra embedding steps. Thus, given a chemical space, we would like to identify sparse regions where there are a few active compounds. Such regions are of interest to researchers as they represent potential regions in which new active compounds may be located. Such analysis is also related to preprocessing the data set using clustering and classification based on *k*NN.

Classification and clustering have been explored extensively, both in the chemical information mining literature and in a multitude of different domains ranging from

* Corresponding author phone: (814) 865-3739; fax: (814) 865-3314; e-mail: pcj@psu.edu.
† University of Southern California.
‡ Pennsylvania State University.
§ These authors contributed equally to this paper.

10.1021/ci050403o CCC: $33.50
Published on Web 11/08/2005

astrophysics to social sciences. A thorough study of the related work is beyond the scope of this paper. A brief survey can be found in a standard book on pattern classification[10,11] or in a recent clustering paper[12] and its references for clustering very large data sets and streams. Commonly used techniques for classification include statistical methods and distance-based methods such as *k*-nearest neighbors,[4] support vector machines, and decision forests.

Thus, the core algorithms for the scalable mining of large chemical data sets can be based on a quick solution to the *k*NN problem. For example, if the *k*NN problem is even approximately solved in *sublinear* time, then one can show that algorithms that have subquadratic running time, in the number of chemical compounds [or $o(n^2)$], are possible for tasks such as clustering and outlier detection. Thus, we would like to have a fast *k*NN routine as the core of a data mining framework.

A simple *k*NN can be done by a linear scan, which makes many mining tasks expensive. For example, outliers would take $\Omega(n^2)$ time. This method can be speeded up using spatial data structures such as kd trees, metric trees, and their variants.[4] Typically such algorithms take $O(\log n)$ per query, on average, for certain distributions of chemical spaces such as the uniform distribution. However, exact guarantees are hard to provide. This makes it especially difficult for mining *dense regions* of chemical space. Also, most optimized versions of *k*NN do not scale well with an increasing number of dimensions because of the well-known *curse of dimensionality*.[13] Thus, we desire a method that is guaranteed to be sublinear in *n* for each *k*NN search operation and which also scales well as the number of dimensions increase. We are willing to trade off accuracy for speed, and we desire approximate methods with coarse-grain guarantees on the accuracy. A guaranteed $o(n)$ method for a *k*NN search can help speed up several data mining applications, as we show in this paper.

Determining the diversity of the chemical space has received a lot of attention in the recent past.[8,14] Most techniques to determine chemical diversity require us to calculate pairwise distances between compounds, and this results in $O(n^2)$ overall complexity. Some attempts have been made to circumvent this bound by using statistical techniques such as the KS test[14] or cell-based methods.[8] We will show, in this paper, that outlier detection can be done by analyzing the nearest-neighbor space around a compound. Again, we conjecture that solving *k*NN efficiently for huge data sets is important.

Note that most of the aforementioned data mining problems do not have guaranteed optimal exact solutions as they are mostly NP-Hard problems,[15] and we need to design either good heuristics or approximation algorithms tuned for the particular data set that needs to be mined. However, mining very large databases, that is, with millions of elements, is an open problem in most application areas and is an active area of research. Such scales are not uncommon in chemical libraries. Our research is based on several goals. First, we want a hierarchical scheme where we can trade off speed for accuracy, thus empowering the user. Next, we want our scheme to be independent of the semantics and the choice of the features we choose to describe the data set.

The core problem we want to solve for very large chemical libraries is the following. Given a large library of *n* compounds, each with their *d* descriptors, represented by a *d* dimensional vector, and a single query point *p*, radius *R*, and error parameter *c*, we want to find all points *q* such that $|p - q| < cR$. We can then use this core subroutine to solve several interesting problems including clustering, classification, outlier detection, and chemical diversity analysis, among several others. In this paper, we focus on the outlier detection problem.

1.1. Our Contributions. Our final goal is to have a single modular framework for the most common tasks in the mining of huge chemical libraries that can be integrated with different informatics pipelines for virtual screening. In this paper, we present a scalable framework for classification and clustering large chemical libraries with the help of recently developed geometric techniques such as random projections that give approximate answers but are guaranteed to run fast.

The basic idea of locality-sensitive hashing (LSH) is to treat each object as a point in high dimensional space and *hash* points so that points close to each other hash to points that are also close in the transformed domain. Hashing is much faster than alternative methods because it avoids the pairwise comparisons required for partitioning and classification. The method especially excels when the data set size is very large. We explore the parameter space and show that there exist parameter subspaces using data sets (libraries) of various sizes that allow LSH to find the nearest neighbors with a high 94% accuracy and answer queries within a fraction of a second even for databases of around 250 000 compounds and 140-dimensional feature vectors.

As an application of this framework, we study a small problem in diversity analysis, that is, the problem of outlier detection. This problem is particularly important if the outliers are active compounds. We can find outliers very quickly in this case, and we have manually validated and confirmed the outliers.

2. METHODS

We now present a very high-level view of our approach. Assume that each chemical compound is described by a set or a feature vector of chemical descriptors. Without a lack of generality, we assume Euclidean descriptors. Given a new compound *x*, its feature vector υ_x, a set of class labels $L = \{l_j\}$, and a very large database $D = \{x_i\}$ of compounds with each x_i having a class label l_i, we assign a label l_x based on its υ_x.

Our approach can be broadly divided into two steps. First, we may optionally want to transform the feature vector into a suitable space. Then, the data is projected on *random* lines using a new algorithm called LSH^2 based on the recently developed idea of random projections.[13] Second, we use these hash functions to detect the approximate nearest neighbors in the high-dimensional chemical descriptor space. The approximate nearest neighbors then form the basis for different mining tasks. In this paper, we use LSH-based nearest-neighbor queries to find outliers quickly.

2.1. Locality-Sensitive Hashing. The basic idea behind random projections is a class of hash functions that are locality-sensitive; that is, if two points (p, q) are close, they will have small $|p - q|$ values and they will hash to the same value with high probability. If they are distant, they should collide with small probability. Thus, we have the following definitions.

Definition 1. A family $\{H = f : S \rightarrow U\}$ is called locality-sensitive if, for any point q, the function

$$p(t) = Pr_H[h(q) = h(v) : |q - v| = t]$$

is strictly decreasing in t. That is, the probability of the collision of points q and v decreases with the distance between them.

Definition 2. A family $H = \{h : S \rightarrow U\}$ is called (r_1, r_2, p_1, p_2)-sensitive for distribution D if, for any v, $q \in S$, we have the following:

- if $v \in B(q, r_1)$, then $\Pr[h(q) = h(v)] \geq p_1$
- if $v \notin B(q, r_2)$, then $\Pr[h(q) = h(v)] \leq p_2$

Here $B(q, r)$ represents a sphere around point q with a radius r. Thus, a good family of hash functions will try to *amplify* the gap between p_1 and p_2.

Indyk et al.[2] showed that s-stable distributions can be used to construct such families of locality-sensitive hash functions. An s-stable distribution is defined as follows.

Definition 3. A distribution D over R is called *s-stable* if there exists s such that, for any n real numbers $v_1 - v_n$ and i. i. d. variables $X_1 - X_n$ with distribution D, the random variable $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i v_i^p)^{1/s} X$, where X is a random variable with distribution D.

Intuitively, consider a random vector a of n dimensions. For any two n-dimensional vectors (p, q), the distance between their projections ($ap - aq$) is distributed as $|p - q|_s X$, where X is an s-stable distribution. We *chop* the real line into equal-width segments of appropriate size and assign hash values to vectorson the basis of which segment they project onto. The above can be shown to be locality-preserving.

There are two parameters to tune LSH. Given a family H of hash functions as defined above, the LSH algorithm chooses k of them and concatenates them to amplify the gap between p_1 and p_2. Thus, for a point v, $g(v) = [h_1(v) - h_k(v)]$. Also, L such groups of hash functions are chosen, independently and uniformly at random, (i.e., $g_1 - g_L$) to reduce the error. During preprocessing, each point v is hashed by the L function's buckets and stored in the bucket given by each of the $g_i(v)$'s. For any query point q, all of the buckets $g_1(q) - g_L(q)$ are searched. For each point x in the buckets, if the distance between q and x is within the query distance, we output this as the nearest neighbor. Thus, the parameters k and L are crucial. It has been shown[2,13] that $k = \log_{1/p_2} n$ and $L = n^\rho$, where $\rho = (\log 1/p_1)/(\log 1/p_2)$, and this ensures locality-sensitive properties. In ref 2, the authors bound ρ, above, empirically by $1/c$, c being the approximation guarantee; that is, for a given radius R, the algorithm returns points whose distances are within $c \times R$. The time complexity of LSH has been shown to be $O(dn^\rho \log n)$, where d is the number of dimensions and ρ is as defined above. Thus, if we desire a coarse level of approximation, LSH can guarantee sublinear run times.

2.2. Exploring Chemical Space. An important problem in the design and analysis of chemical libraries is the determination of which regions of the library are underrepresented. That is, we are interested in sparse regions of the chemical space defined by a set of descriptors. An answer to this type of question would allow the users of a library to understand what types of compounds can be acquired to better represent a region of chemical space. The question can be reversed by asking which compounds lie in the denser regions of chemical space covered by the library. An answer to this question allows the user to be able to select similar groups of compounds from differing regions of the space. Clearly, by combining both questions, one may gain an understanding of which compounds to focus on during either the compound acquisition or compound selection steps. The LSH algorithm provides a robust approach to answering both of the above questions.

Recall that the LSH algorithm considers a user-specified radius and then generates a list of approximate nearest neighbors for each query point in sublinear time. Thus, by successively increasing the radius in some fashion, discussed later, one may include an increasing number of approximate nearest neighbors. A side effect of this approach is that, while increasing the radius, certain compounds might have only one or two nearest neighbors, whereas others will have hundreds (or even thousands). Intuitively, this situation indicates that singleton or near-singleton compounds are located in very sparse regions of the chemical space, and thus, for increasing radii, they will not have a significant number of nearest neighbors. In comparison, using the traditional kNN algorithm, these singleton or near-singleton compounds will always have a set of nearest neighbors. In this case, determining whether a compound exists in a sparse region of the space would require an analysis of the distribution of nearest-neighbor distances, which is clearly time-consuming. On the other hand, using the LSH algorithm, one simply has to supply a query point. It will be shown that the query time for a single point is extremely small.

Thus, a simple algorithm for obtaining outliers is to run the LSH algorithm for each point and for increasing radii iteratively and check whether the number of points exceeds a chosen *sparsity threshold*. If so, the region is not sparse. For n points, the traditional kNN will take $O(dn^2)$, whereas LSH will take $O(dn^{\rho+1} \log n)$, which is clearly $o(n^2)$ or, in other words, less than $O(n^2)$. Though we use approximate nearest neighbors, we do not lose accuracy in finding sparse regions. In this case, if a region within a sphere of radius $r(1 + \epsilon)$ is sparse, where ϵ is due to the approximation, it is sparse within the sphere of radius r too. The natural question that arises from the above algorithm is how to vary the query radius. A naive way is to increase it linearly. A better approach is to use a technique similar to a binary search, which will take $\log D_m$ steps of LSH, where D_m is the maximum distance specified.

2.3. Data Sets. To test the approach, we considered three data sets. The first data set contained 4337 compounds and was described by Kazius et al.[16] The molecules in this data set were studied using the AMES test.[17] The second data set consisted of 42 689 compounds[18,19] from the NCI repository. The compounds were studied using the DTP AIDS antiviral screen and were classified as confirmed active, confirmed moderately active,or confirmed inactive. The structures were 2D and were converted to 3D and washed using MOE[20] with the MMFF94 force field with a 0.1 Å tolerance. However, a few molecules could not be converted successfully to 3D and, thus, were removed from the data set, resulting in a final data set consisting of 42 613 compounds. The third data set was also obtained from the NCI repository and contained 249 071 molecules.[19,21] The

Table 1. Summary of the Data Sets Used in This Study[a]

data set	number of molecules	number of descriptors	
		full	reduced
Kazius[16]	4337	142	20
NCI−AIDS	42 613	143	55
NCI−3D	24 9071	122	52

[a] The column titled full indicates the total number of descriptors calculated, and the column titled reduced indicates the size of the descriptor pool after reduction using correlation and identity testing.

Table 2. Summary of the Mean and Maximum Pairwise Distances in the Different Data Sets, Using the Full and Reduced Pools of Descriptors for Each Data Set[a]

data set	maximum distance		mean distance	
	full	reduced	full	reduced
Kazius[16]	507 252	507 144	1793	1656
NCI−AIDS	626 517	626 438	3712	3490
NCI−3D	14 285 759	3 013 170	10 810	11 784

[a] All distances were rounded to the nearest integer. For the NCI data sets, the mean and maximum pairwise distances were obtained by randomly selecting 10% of the data set four times.

structures contained 3D coordinates which were washed and optimized using MOE using the MMF94 force field with a 0.1 Å tolerance. No property was available for this data set.

For all three data sets, we evaluated MACCS fingerprints[22] and calculated descriptors using the MOE software package. We calculated topological and geometric descriptors, ignoring semiempirical electronic descriptors because of the long time required for calculation. Table 1 summarizes the number of compounds and the number of descriptors calculated for each data set. The topological descriptors included the Wiener path index,[23] the Zagreb index, Balaban's *J* topological index,[24,25] Kier shape descriptors,[26−28] and the zeroth- and first-order χ indices.[29−31] In addition, a number of constitutional descriptors such as counts of heavy atoms, bonds, and hydrogen-bond donors and acceptors were also evaluated. Geometric descriptors included the van der Waals surface area and volume descriptors and density. A number of hybrid descriptors were also evaluated. These included charged polar surface areas (using the Gasteiger−Marsilli charges[32,33]) as well as the topological polar surface[34] area descriptor. The octanol−water partition coefficient was also evaluated.

We also investigated the effect of descriptor reduction on the results of the LSH algorithm. The original pool of descriptors for each data set was reduced in two steps. First, identical testing was carried out, whereby descriptors which were constant for more than 70% of the observations were discarded. Next, a correlation test was carried out in which the pairwise correlation coefficient was evaluated and, for pairs having a correlation greater than 0.6, one member of the pair was randomly selected and discarded. Table 2 summarizes the maximum and mean pairwise distances for the full and reduced pool of descriptors for each data set. For the NCI data sets, we obtained the maximum and mean distances by a random sampling procedure rather than a full analysis of the complete pairwise distance matrix.

The LSH calculations were based on C++ code from Professor Indyk, and the traditional *k*NN algorithm was examined using the implementation in the R software package.[35] It should be noted that, in the latter case, the algorithm was coded in C and R was only used as a front-end to the C routine. As a result, R simply called the precompiled *k*NN routine, and consequently, none of the timings include the time taken to load the data into memory. All calculations were performed on a 2.0 GHz AMD Opteron processor with 4 GB of ECC RAM and standard SATA hard drives running Fedora Core 3. We also tested the performance on other machines including Dothan laptops and Intel P4 desktop processors, and the results were similar.

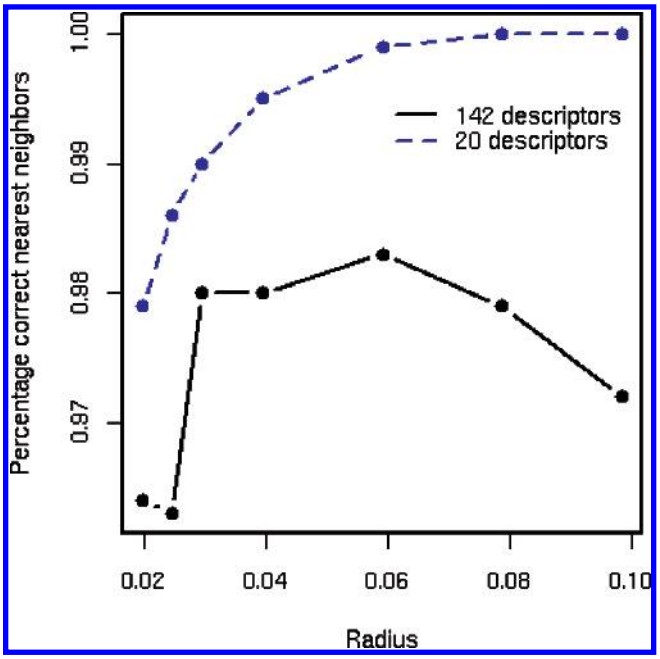

Figure 1. Plot showing the percentage of correct *R* nearest neighbors detected by the LSH algorithm versus the radius for the Kazius data set using the original and reduced descriptor pools. The radii are reported as a percentage of the maximum pairwise distance in the original descriptor pool.

3. RESULTS

3.1. Accuracy. First, we validate the accuracy of our framework. For this purpose, we focused on the smaller data sets, the Kazius and the NCI−AIDS. With these data sets, we varied the radius and compared the outputs of the approximate *k*NN algorithm as well as the exact *k*NN algorithm. Our accuracy was at least 94%. The accuracy results for the Kazius data set are shown in Figure 1. In Figure 2, we plot the accuracy results for the NCI−AIDS data set, while Figure 3 compares the accuracy for the NCI−3D data set. We then considered different variations of the data sets with reduced descriptors, and the accuracy was higher. Observe that the accuracy has not dropped with the size of the data set. In fact, the minimum accuracy for the NCI−3D data set is 95%, which is more than that of the smaller NCI−AIDS data set.

These results clearly demonstrate that the core nearest-neighbor approximate algorithm is indeed accurate for a virtual screening procedure. Our accuracy results were consistent across radii and the number of nearest neighbors for different radii. It is important to note that the results for the smaller radii are more critical than those at higher radii. In all of the data sets we analyzed, the accuracy dipped and went up back again as we increased the radii because a

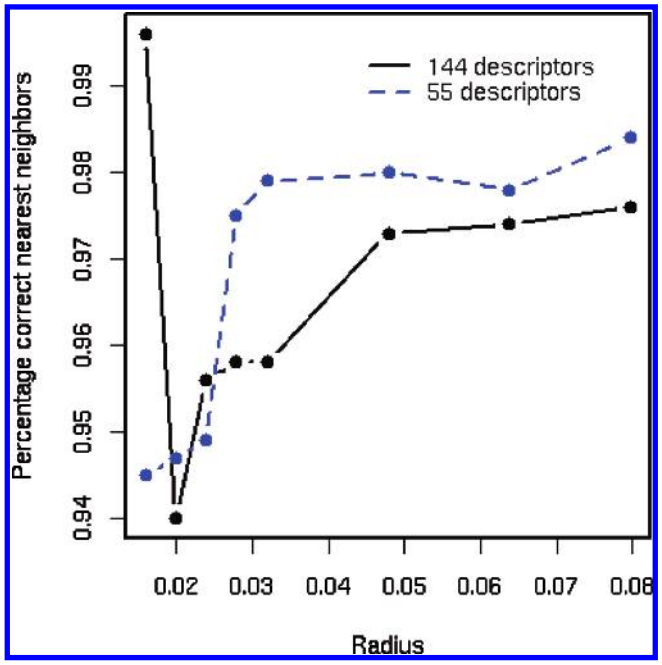

Figure 2. Plot showing the percentage of correct R nearest neighbors detected by the LSH algorithm versus the radius for the NCI−AIDS data set using the original and reduced descriptor pools. The radii are reported as a percentage of the maximum pairwise distance in the original descriptor pool.

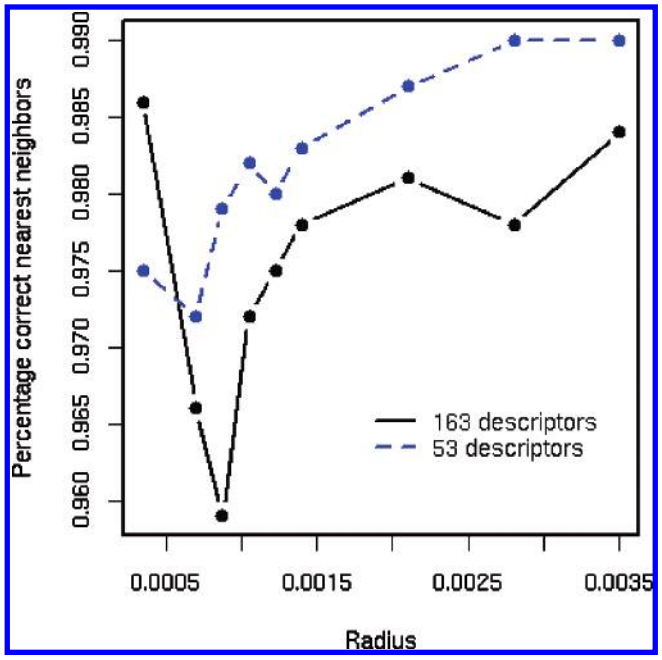

Figure 3. Plot showing the percentage of correct R nearest neighbors detected by the LSH algorithm versus the radius for the NCI−3D data set using the original and reduced descriptor pools. The radii are reported as a percentage of the maximum pairwise distance in the original descriptor pool.

smaller radius forces the algorithm to probe denser regions. With larger radii, the number of nearest neighbors is also large and accuracy increases.

It is well-known that, even for the nearest-neighbor classifier, the error is at most twice the Bayesian classification error. Hence, our framework can be useful for the classification of large data sets as it will, at most, add a factor of 0.94 to any other classifier based on *k*NN. A thorough investigation of this is beyond the scope of this paper and is underway.

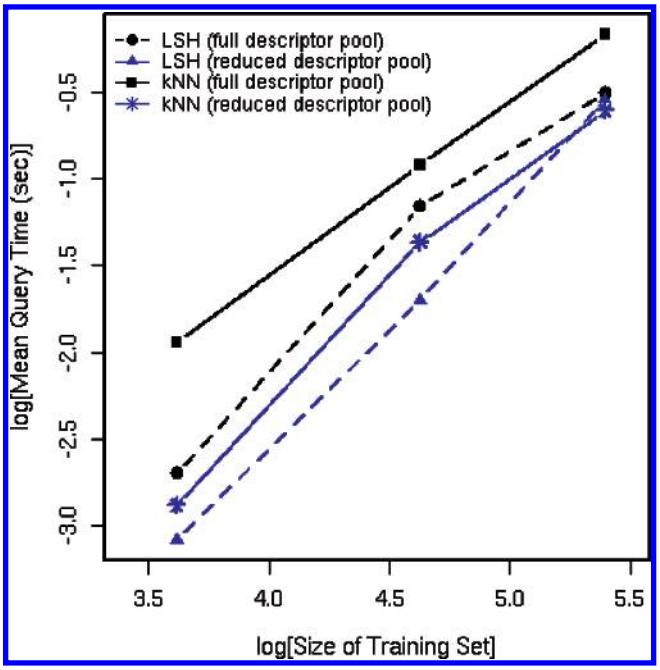

Figure 4. Plot of mean query times versus the size of the data set plotted on a log−log scale using both original and reduced descriptor pools. For each data set, the first 200 observations were taken as the query set and the remainder placed in the training set. In each case, the radius was set to 0.1 of the maximum pairwise distance in the data set. For comparison, the mean query times for the traditional *k*NN with $k = 200$ are also plotted.

Here, our goal is to introduce a framework based on near neighbors and demonstrate its use in exploring the chemical spaces, as discussed later in this section.

3.2. Execution Times. We were concerned with how fast our framework would be in comparison with a standard *k*NN routine with a large number of nearest neighbors. This is because, in applications such as chemical diversity analysis, we might need to explore regions in the chemical space that are quite dense, and determining all the compounds within a given radius might yield hundreds of neighbors, as our plots clearly show. Thus, we chose $k = 200$ arbitrarily. Also, from our data sets, we chose the first 200 compounds arbitrarily as the query set. We tried other schemes (including using the whole training set as the query set), and the results were similar.

For each data set, we find approximate nearest neighbors for both the full descriptor pool and the reduced descriptor set to test the effect of the number of descriptors or features. Our primary metric for comparison is the mean query time. For example, when we used the largest data set, that is, the NCI−3D data set of 250 000 compounds, the mean query time was 0.0002 s for LSH using a small radius that yielded around five nearest neighbors and was up to 0.03 s for radii that yielded more than 20 000 neighbors. For radii that yielded around 200 neighbors, the mean query time was 0.004 s. On the other hand, *k*NN took 0.665 s per query. We computed a modest 200 nearest neighbors per query. Thus, we achieved a speedup of up to 3 orders of magnitude for a similar number of nearest neighbors. For practical scenarios, the speedup is around 2−3 orders of magnitude. We show similar speedups in Figure 4. For the reduced descriptor set, the speedups were not as great. Thus, LSH seems to be relatively faster for higher dimensionalities

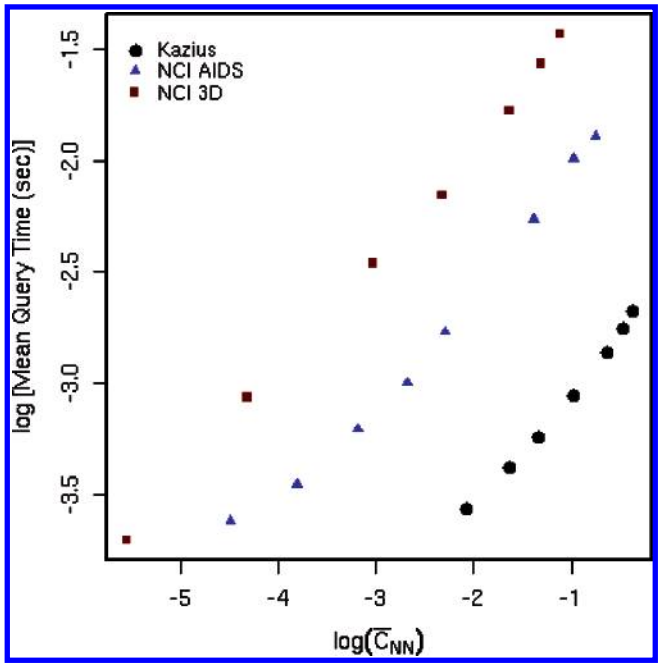

Figure 5. Plot of the mean query time versus the normalized mean count of nearest neighbors per query point. Note that both axes are logarithmic, indicating that the mean query time increases exponentially with increasing number of nearest neighbors detected. For each data set, 200 observations were placed in the query set.

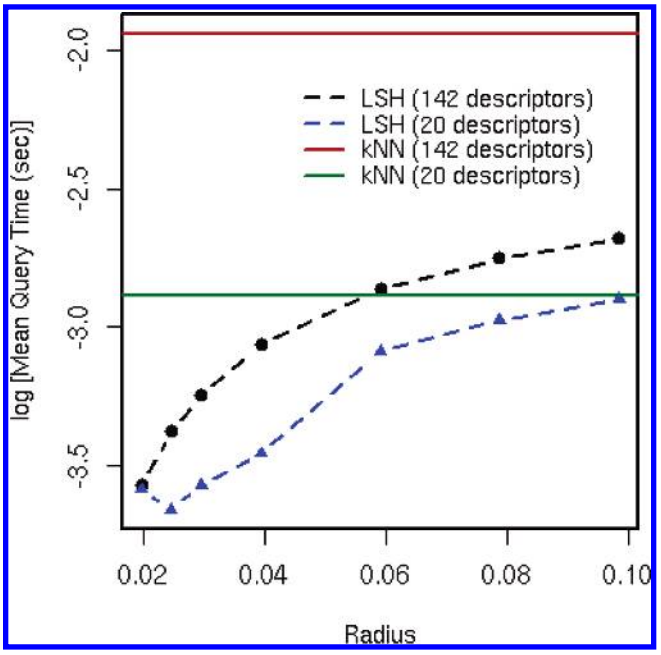

Figure 6. Plot of the mean query time versus the radius for a 200-observation query set taken from the Kazius data set. The radii are reported as a percentage of the maximum pairwise distance in the data set, and the results are shown for both the original and reduced descriptor pools. For comparison, the mean query times for the same query set using the *k*NN algorithm ($k = 200$) are plotted.

across the three data sets. We also observed (in Figure 5) a monotonically increasing relationship between the logarithms of the mean query time and that of the number of neighbors that LSH found. One possible explanation for this observation is that, in order to yield several neighbors, it is the memory access that becomes the bottleneck. In Figures 6−8, we always see a speedup of around 2 orders of magnitude over the *k*NN classifier across the three data sets we tried. However, the speedup, in practice, can even be 3−4 orders of magnitude, as shown in the figures.

Note that the *total execution time* has not been used for our comparisons for the following reasons. LSH has a parameter tuning step which needs to be done once for each database before all the queries. This is at most 2−3 min for the largest data sets we have tried (the NCI−3D data set with about 250 000 compounds). *k*NN does not need to do this step. In fact, LSH excels when there is a need for a large number of nearest-neighbor queries. If we need to find answers to a few queries, then a simple linear scan of all the data will also be acceptable. Then, we compare the mean query time by dividing the total query time by the number of queries.

We also feel the need to use another metric called the *normalized query time* and to compare the running times of LSH with that of traditional *k*NN. This is given by the query time for a particular query divided by the number of neighbors. For very large chemical libraries, the number of neighbors might be very large. For a large number of neighbors, the time taken to retrieve the data points might require several memory accesses. It is well-known that, for processing huge data sets, memory access time dominates the processing time, and it is a few orders of magnitude larger than the latter. We found a monotonically superlinear increasing relationship between the query time and the number of nearest neighbors reported by LSH. Note that, in

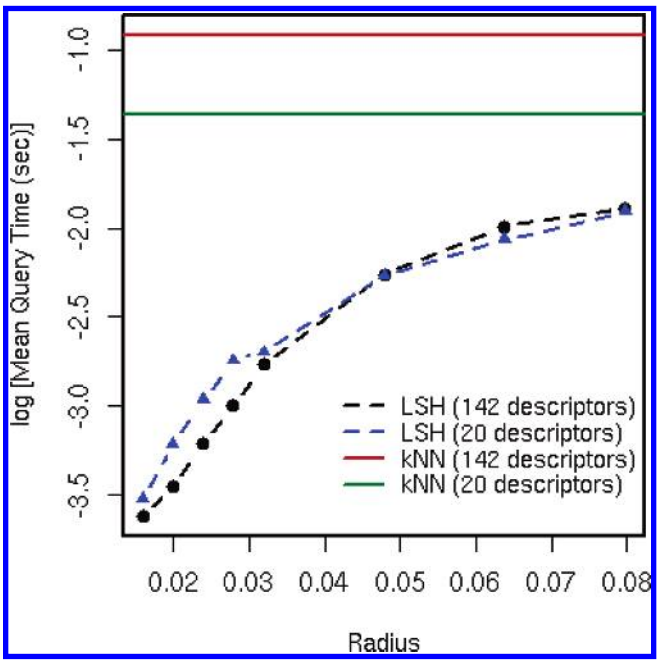

Figure 7. Plot of the mean query time versus the radius for a 200-observation query set taken from the NCI−AIDS data set. The radii are reported as a percentage of the maximum pairwise distance in the data set, and the results are shown for both the original and reduced descriptor pools. For comparison, the mean query times for the same query set using the *k*NN algorithm ($k = 200$) are plotted.

Figure 4, the scales are all logarithmic. This shows that our metric is reasonable. Using this metric, it is possible to show that the LSH algorithm is about 4 orders of magnitude faster than the traditional *k*NN for the data sets we have considered.

Another point to note is that LSH will perform better (faster) with increased numbers of descriptors. Consider a

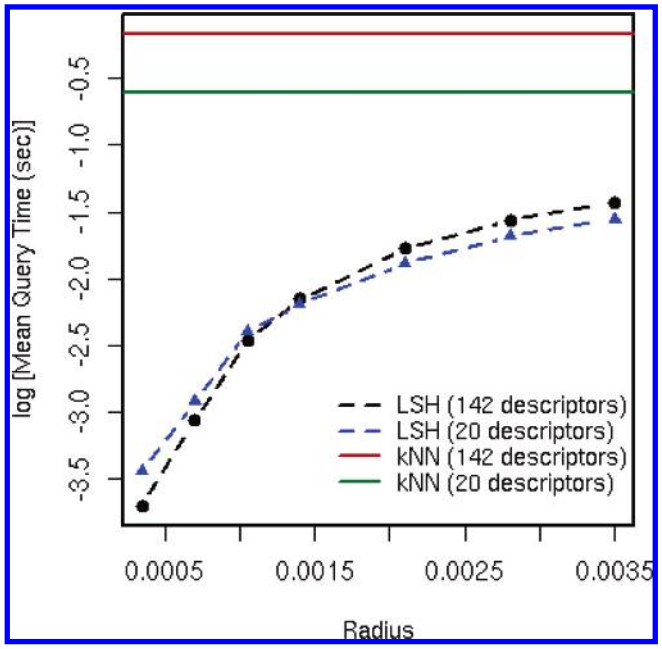

Figure 8. Plot of the mean query time versus the radius for a 200-observation query set taken from the NCI−3D data set. The radii are reported as a percentage of the maximum pairwise distance in the data set, and the results are shown for both the original and reduced descriptor pools. For comparison, the mean query times for the same query set using the *k*NN algorithm ($k = 200$) are plotted.

scenario when the number of dimensions is $d = O(n)$. For a traditional *k*NN, each distance calculation now takes $O(d) = O(n)$ time for *each* point that it compares with. On the other hand, in LSH, we need to do a few dot products (depending on the parameters that dictate the number of hash functions chosen) per query.

3.3. Effects of the Data Set Dimension. Theoretically, the LSH algorithm is linear in the number of dimensions of the data set. This is an attractive feature as it saves us from having to perform objective feature selection. Traditionally, this type of feature selection has involved removing descriptors which are identical for a certain percentage of the data set (identical test) and removing correlated descriptors (correlation test). Though the former task is quite rapid, the latter can be time-consuming for larger data sets. We, thus, investigated whether the LSH algorithm would be resistant to correlated descriptors. We performed a descriptor reduction using an identical test with a cutoff set to 0.7 and a correlation test with a cutoff set to 0.6. The sizes of the reduced descriptor pools for the three data sets are summarized in Table 1.

Figures 6−8 plot the mean query time for a query set of 200 observations versus the radius. The radii are reported as a percentage of the maximum pairwise distance in the data using the original descriptor pool. For the Kazius data set, the mean query time for the reduced descriptor pool is generally smaller than that for the original descriptor pool. Since the *y* axis is plotted on a logarithmic scale, the speedup is approximately two to three times that when a smaller descriptor pool is used to perform LSH. For comparison, the mean query times for the *k*NN algorithm using the original and reduced descriptor pools are plotted. The *k*NN algorithm was run with $k = 200$, as described before. It is clear that, by using the reduced descriptor pool, the mean query time for the *k*NN algorithm increases by nearly 1 order of magnitude. In comparison, with increasing radii, the speed of the LSH algorithm using the reduced descriptor pool approaches that of the traditional *k*NN algorithm. Considering the results for the NCI−AIDS data set shown in Figure 7, we see that the use of the reduced pool does not lead to a significant improvement in mean query time, and in fact, for smaller radii, the mean query time for the reduced descriptor set is worse than that for the full descriptor set. However, in both cases, the mean query times are 1 order of magnitude and, for smaller radii, 2 orders of magnitude faster compared to the *k*NN algorithm ($k = 200$). Furthermore, given the fact that the decrease in mean query time for the *k*NN runs using the reduced descriptor pool is not as large compared to that of the Kazius data set, it is not surprising that the LSH algorithm does not show a significant improvement in mean query time when using the reduced descriptor pool. Figure 8 displays the results for the NCI−3D data set, and it is clear that the behavior is similar to that of the NCI−AIDS data set. In this case, however, the mean query time for the reduced descriptor pool is lower for a larger set of radii compared to the NCI−AIDS data set. It is clear that, for both the original and reduced descriptor pools, the mean query time for the LSH algorithm is significantly lower compared to the *k*NN algorithm.

We were also interested in understanding how the radius affects the number of nearest neighbors detected for a given point. Intuitively, we expect that, as the radius increases, the number of nearest neighbors for a query point should increase. Thus, we considered a metric termed the normalized mean count of nearest neighbors per query point and investigated its variation with radius. This metric is defined as

$$\bar{C}_{\mathrm{NN}} = \frac{1}{N_{\mathrm{t}}}\frac{1}{N_{\mathrm{q}}}\sum_{i=1}^{N_{\mathrm{q}}} N_{\mathrm{NN},i}$$

where N_{t} and N_{q} are the number of points in the training and query sets, respectively, and $N_{\mathrm{NN},i}$ is the number of nearest neighbors for the *i*th query point. Essentially, this metric measures the number of nearest neighbors per query point normalized by the size of the training set. Considering the variation of $\bar{C}_{\mathrm{NN}}$ for the Kazius data set (Figure 9), we see that the count is consistently larger when the reduced descriptor pool is used. This is a useful feature since it allows one to use a smaller radius but still obtain a sufficient number of nearest neighbors for a given query point. However, this cannot be generalized completely to other data sets. As shown in Figures 10 and 11, there is no significant difference in the nearest-neighbor count at the lower radii. However, after a certain point, in both cases, the mean nearest-neighbor count increases significantly with the radius for the reduced descriptor pools. For example, if we consider the NCI−AIDS data set, we see that, for a radius of approximately 0.05% of the maximum pairwise distance in the data set, the mean number of nearest neighbors per query point is 1755 and 4456 for the original and reduced descriptor pools, respectively. It is, thus, clear that, if one were to use LSH as a partitioning procedure, it would be advantageous to use the reduced descriptor pool, so that a smaller radius could be used to achieve lower mean query times.

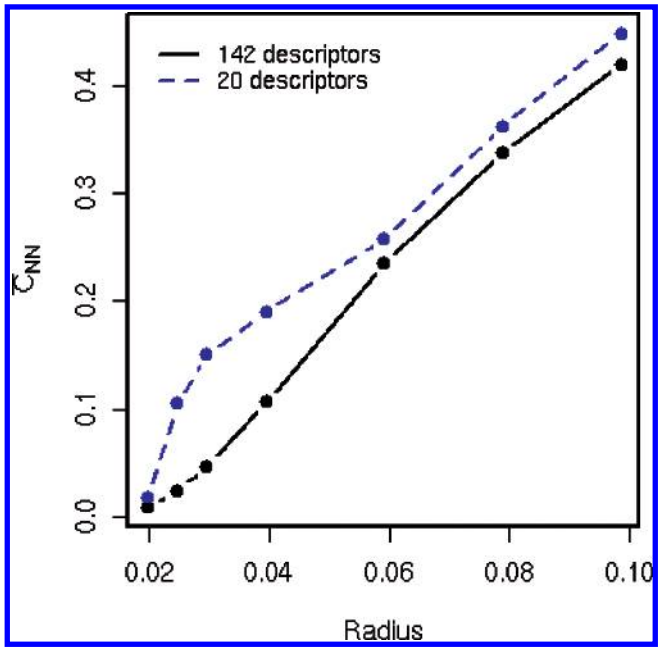

Figure 9. Normalized mean count of NN per query point for varying radii, for the Kazius data set using the original and reduced descriptor pools. For both cases, the query set consisted of 200 observations and the remainder were placed in the training set. The radii are reported as a percentage of the maximum pairwise distance in the original descriptor pool.

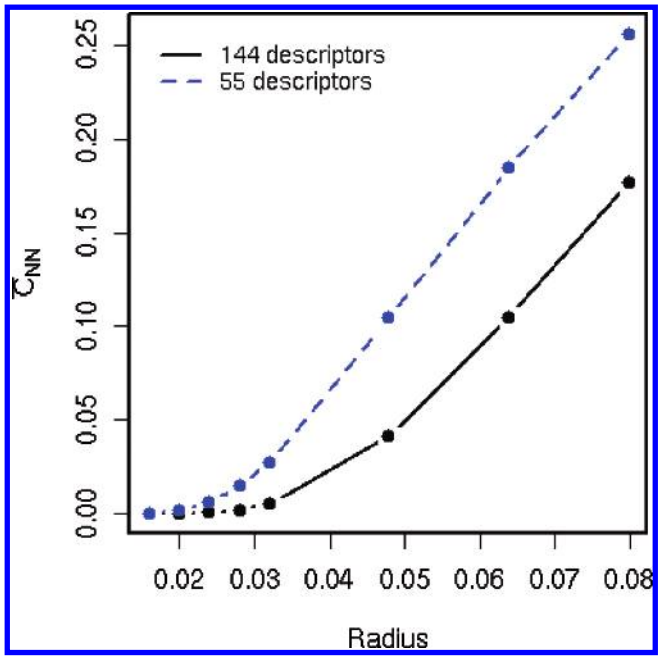

Figure 10. Normalized mean count of NN per query point for varying radii, for the NCI−AIDS data set using the original and reduced descriptor pools. For both cases, the query set consisted of 200 observations and the remainder were placed in the training set. The radii are reported as a percentage of the maximum pairwise distance in the original descriptor pool.

Similar results were observed for the NCI−3D data set. In fact, for this data set, it was observed that, even for a radius equal to 0.0035% of the maximum pairwise distance in the data set, the value of $\bar{C}_{NN}$ ranged from 0.07 to 0.09. That is, on average, there were 17 434−22 416 nearest neighbors per query point depending on whether the original or reduced descriptor pool was used. As in the case of the other data sets, the reduced data set leads to a higher value of $\bar{C}_{NN}$ for larger radii. In this case, given the large number of nearest neighbors per query point for both the original and reduced descriptor pools, descriptor reduction is not necessary if all we require is a large set of nearest neighbors for further investigation.

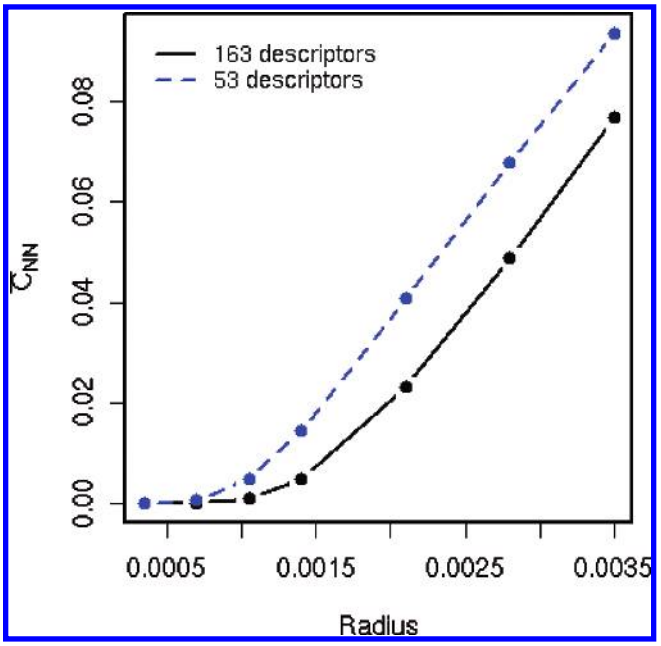

Figure 11. Normalized mean count of NN per query point for varying radii, for the NCI−3D data set using the original and reduced descriptor pools. For both cases, the query set consisted of 200 observations and the remainder were placed in the training set. The radii are reported as a percentage of the maximum pairwise distance in the original descriptor pool.

Apart from the mean number of nearest neighbors per query point, we must also consider the accuracy of the nearest neighbors. Figures 1−3 summarize the percentage of nearest neighbors that the LSH algorithm detected versus the radius compared to the nearest neighbors detected by the exact algorithm using a linear scanning procedure. In the case of the Kazius data set (Figure 1), we see that using the reduced descriptor set results in a consistent increase in the percentage of nearest neighbors that the LSH algorithm detects correctly. In the case of the original descriptor pool, the accuracy is not consistent and appears to decrease at higher radii. However, we also considered radii between 10% and 50% of the maximum pairwise distance, which resulted in up to 90% of the data set being considered as nearest neighbors for a given query point. At these radii, the nearest neighbors detected by the LSH algorithm were identical to those detected by the exact algorithm. If we then consider the NCI−AIDS data set, we see that the variation in accuracy with the radius is more erratic, though as before the use of the reduced pool does lead to consistently higher accuracies. As with the Kazius data set, using radii between 10% and 50% of the maximum pairwise distances resulted in the LSH algorithm performing with 100% accuracy. In the case of the NCI−3D data set, we find a similar situation. The lowest accuracy was 96%, and in general, there is an increasing trend with respect to the radius employed. As with the other data sets, the accuracy obtained using the reduced pool of descriptors is consistently higher. Interestingly, in the case of this data set, one does not need to use a very large radius

Table 3. Summary Statistics of the 1-Nearest Neighbor Distances for Each Observation in the Kazius Data Set[a]

algorithm	number of descriptors	mean	standard deviation
LSH	142	45.43	26.04
	20	21.24	15.88
*k*NN	142	45.42	26.03
	20	21.23	15.88

[a] The nearest neighbors were obtained from the LSH algorithm with the radius equal to 0.06% of the maximum pairwise distance in the data set. For comparison, the summary statistics for the nearest neighbor distances obtained by the 1-NN algorithm are also presented.

to reach 100% accuracy. In Figure 3, we see that, at 0.0035% of the maximum pairwise distance, the data set using the reduced descriptor pool produces results with 99% accuracy. We observed that using radii beyond 10% of the maximum pairwise distance resulted in 100% accuracy for this data set.

The above discussion indicates that, in general, the use of a reduced descriptor pool provides some advantages in terms of mean query times and the mean number of nearest neighbors detected per query point. However, given that using the original descriptor pool results in mean query times that are generally 1 order of magnitude faster compared to those of the *k*NN algorithm, is there any advantage in performing descriptor reduction? This may be answered by considering the statistics of the 1-NN distances for each query point. In the case of the LSH algorithm, each query is associated with a set of nearest neighbors. Thus, for each query point, we considered the nearest neighbor in the set that was closest to the query point. For all three data sets and for all query points in a given data set, the nearest neighbor detected was identical to the nearest neighbor obtained by the *k*NN algorithm ($k = 1$). We then evaluated the mean and standard deviations of the distances from each query point to its nearest neighbor as detected by both algorithms. The results of this calculation for the Kazius data set are summarized in Table 3. The results for the LSH algorithm were obtained at a radius of 0.06% of the maximum pairwise distance in the data set. As a result of the low radius, a few query points had themselves as the nearest neighbors, and these query points were excluded from the *k*NN calculations as well when calculating the summary statistics. From Table 3, it is clear that the statistics of the 1-NN distances are identical for both the *k*NN and LSH algorithms. In general, performing a descriptor reduction leads to better results when using the *k*NN algorithm for classification or regression. Clearly, even though the LSH algorithm is significantly faster than the *k*NN algorithm using either the original or reduced descriptor pool, it is advisable to use a reduced descriptor pool to achieve levels of classification or regression performance similar to those obtained using the *k*NN algorithm.

3.4. Exploring Chemical Space. As mentioned previously, the LSH algorithm can be used to explore a descriptor space for a set of molecules. We investigated the use of the LSH algorithm to detect outlying molecules in the Kazius data set. The strategy consisted of running both phases of the LSH algorithm (parameter tuning and query) for the whole data set, with radii varying from 0.1% to 25% of the maximum pairwise distance in the data set. It should be noted that, unlike the previous experiments, we considered the whole data set as the query set rather than taking a subset of the whole data set. We considered both the original pool of 142 descriptors and the reduced pool of 20 descriptors. For a given set of descriptors, we ran the LSH algorithm for successively increasing radii. We then noted, for each radius, which observations were regarded as singletons, that is, observations for which the nearest neighbor was itself. Our premise is that those observations which are regarded as singletons for successively increasing radii can be regarded as occupying sparse regions of the chemical space defined by the data set and descriptors used. Though we have only considered singletons, one may relax the condition for sparsity of chemical space by also considering compounds that have only a few (say two to five) nearest neighbors. These compounds and their nearest neighbors would constitute an isolated cluster of compounds in a relatively unoccupied region of the chemical space. Depending on the rate of change of the count of nearest neighbors for a given query point, it is possible to understand the density of the region containing the point in question in a qualitative manner.

Running the LSH algorithm on the Kazius data set using 142 descriptors resulted in 27 observations marked as singletons when the radius was set to 0.1% of the maximum pairwise distance. In the case of the reduced pool of descriptors for the same data set, the algorithm detected 15 singleton observations for the same radius. With decreasing radii, the number of singletons detected was observed to increase significantly. This behavior is not surprising as the use of a smaller radius necessarily reduces the number of nearest neighbors.

One approach to visualizing the results is to plot the first principal component versus the second principal component of the data set. The resultant plot provides one view of data with respect to the components being plotted. Figure 12 shows a plot of the first principal component versus the second principal component, which together explained 99.97% of the total variance in the Kazius data set when using 142 descriptors. The points marked in blue represent observations that were marked as singletons, as described above. It is clear that, for the given components, the majority of singletons correspond to isolated compounds and compounds relatively distant from the bulk of the data set. However, a number of points lie near the bulk of the plot (upper-right region). This might indicate that these points are not really outliers. When other principal components are considered, these points do indeed lie in sparse regions of those plots. We observed a similar situation when we considered the reduced descriptor pool of 20 descriptors. Figure 13 displays a plot of the first principal component versus the second principal component (which together explain 99.99% of the total variance in this data set). In this case, the number of singletons detected is lower than when using all of the descriptors. However, most of the singletons detected do lie away from the bulk of the plot. As before, points that appear to lie in or close to the bulk are more isolated when other pairs of components are considered. The singletons detected by the LSH algorithm using the reduced descriptor pool were a subset of the singletons detected by the algorithm using the original descriptor pool. From these plots, one may conclude that large descriptor pools do not

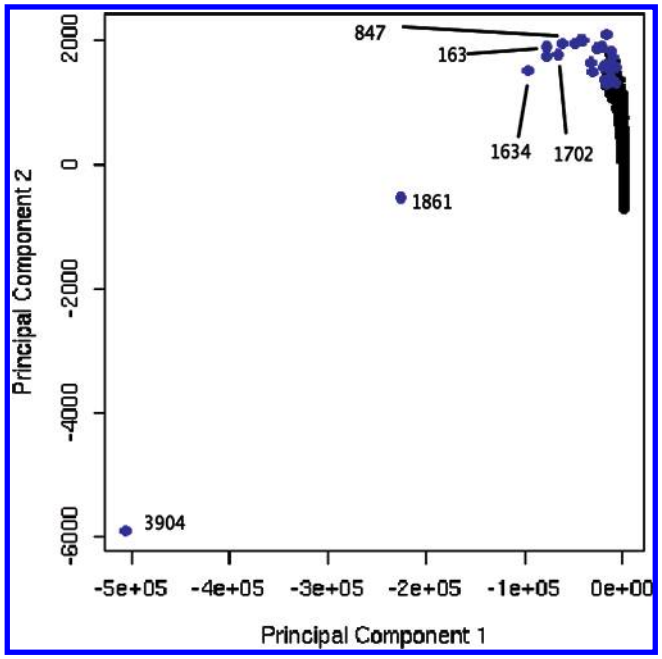

Figure 12. Plot of the first versus second principal component of the Kazius data set, using 142 descriptors. Points marked in blue represent singleton observations detected by the LSH algorithm with the radius set to 0.1% of the maximum pairwise distance in the data set.

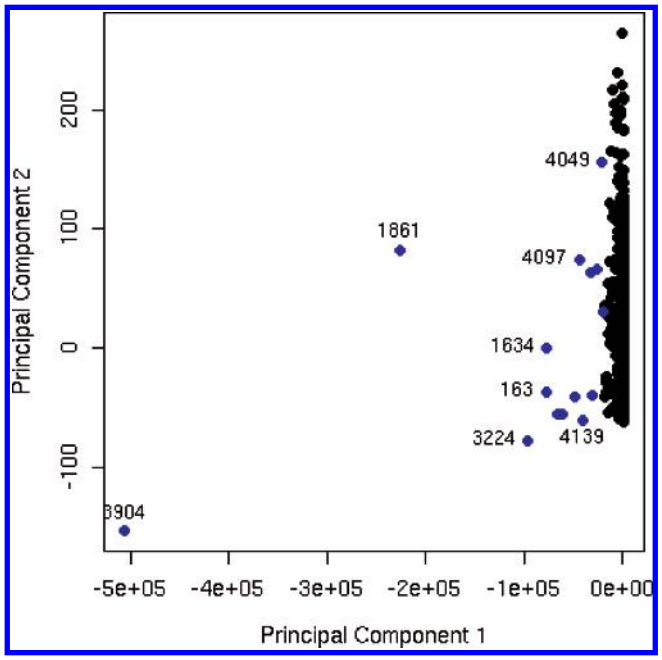

Figure 13. Plot of the first versus second principal component of the Kazius data set, using 20 descriptors. Points marked in blue represent singleton observations detected by the LSH algorithm with the radius set to 0.1% of the maximum pairwise distance in the data set.

necessarily affect the ability of the algorithm to detect singleton observations. That is, descriptor reduction is not a requirement. Table 4 summarizes the mean distance from the singletons annotated in Figure 13 to all the other members of the data set, using the original and reduced descriptor pools. It is clear that, in terms of Euclidean distance, the singletons do indeed lie far from the bulk of the points, confirming that the LSH algorithm is able to reliably detect points lying in sparse regions of descriptor space.

Table 4. Summary of the Mean Euclidean Distances from Each of the Singleton Observations in Figure 13 to the Rest of the Kazius Data Set[a]

	mean distance		
index number	full data set	reduced data set	AMES result
163	78 341.42	78 221.16	mutagen
1634	77 187.24	77 075.04	nonmutagen
1861	226 499.30	226 409.20	mutagen
3224	96 673.50	96 566.88	mutagen
3904	506 239.70	506 146.40	nonmutagen
4049	21 576.62	21 423.29	nonmutagen
4097	44 776.78	44 644.02	nonmutagen
4139	40 916.20	40 785.37	nonmutagen

[a] The full data set refers to the original 142-member descriptor pool, and the reduced data set corresponds to the reduced 20-member descriptor pool.

By successively increasing the radii for the LSH algorithm, we can focus on more isolated compounds. Thus, Figures 12 and 13 both highlight singletons detected at a specific radius. By successively increasing the radius, we can focus on points that are located in sparser regions of the descriptor space. Our experiments indicated that, when the radius was set to more than 1% of the maximum pairwise distance, the LSH algorithm detected the same number of singletons, with either the original or reduced descriptor pool. Furthermore, the singletons detected were the same. For radii beyond 5% of the maximum pairwise distance, the algorithm detected only two singletons. These correspond to points **3904** and **1861** in Figure 13, in which it is clear that these two points lie far from the bulk of the data set. Figures 14 and 15 show the structures of these outliers. The structures are quite distinct from the bulk of the data set, which consists of relatively smaller molecules. It is also interesting to note that molecule **3904**, which is the most outlying point in the principal component plots as well, in terms of Euclidean distance, is classified as a nonmutagen (Table 4). Since this molecule is one the singletons consistently identified with increasing radii, this region of the chemical space is sparse and, hence, it could be beneficial to use molecule **3904** as the starting point for a scaffold-hopping approach to further explore this region for other nonmutagenic compounds. Some examples of compounds drawn from the denser regions of the chemical space (identified by their having a large number of nearest neighbors) are shown in Figure 16. We also calculated the average Tanimoto similarity between the two outliers noted above with the remainder of the data set, using MACCS fingerprints. However, in both cases, the average similarity was not significantly different from the average Tanimoto similarity for the whole data set. One reason for this behavior is that the data set did contain some larger molecules and, as a whole, the data set was quite heterogeneous. Furthermore, the two outliers that were highlighted by the LSH algorithm would contain a number of features in common with smaller molecules. As a result, the fingerprints would be expected to contain a number of bits that would be set to 1 in both the outliers as well as in the nonoutliers.

The above discussion indicates that, by varying the radius, we can zoom out to successively sparser regions of the descriptor space defined by the data set. In the limiting radius (i.e., equal to the maximum pairwise radius), there will be no query point that will be a singleton. Thus, a possible strategy to look for isolated points is to consider relatively

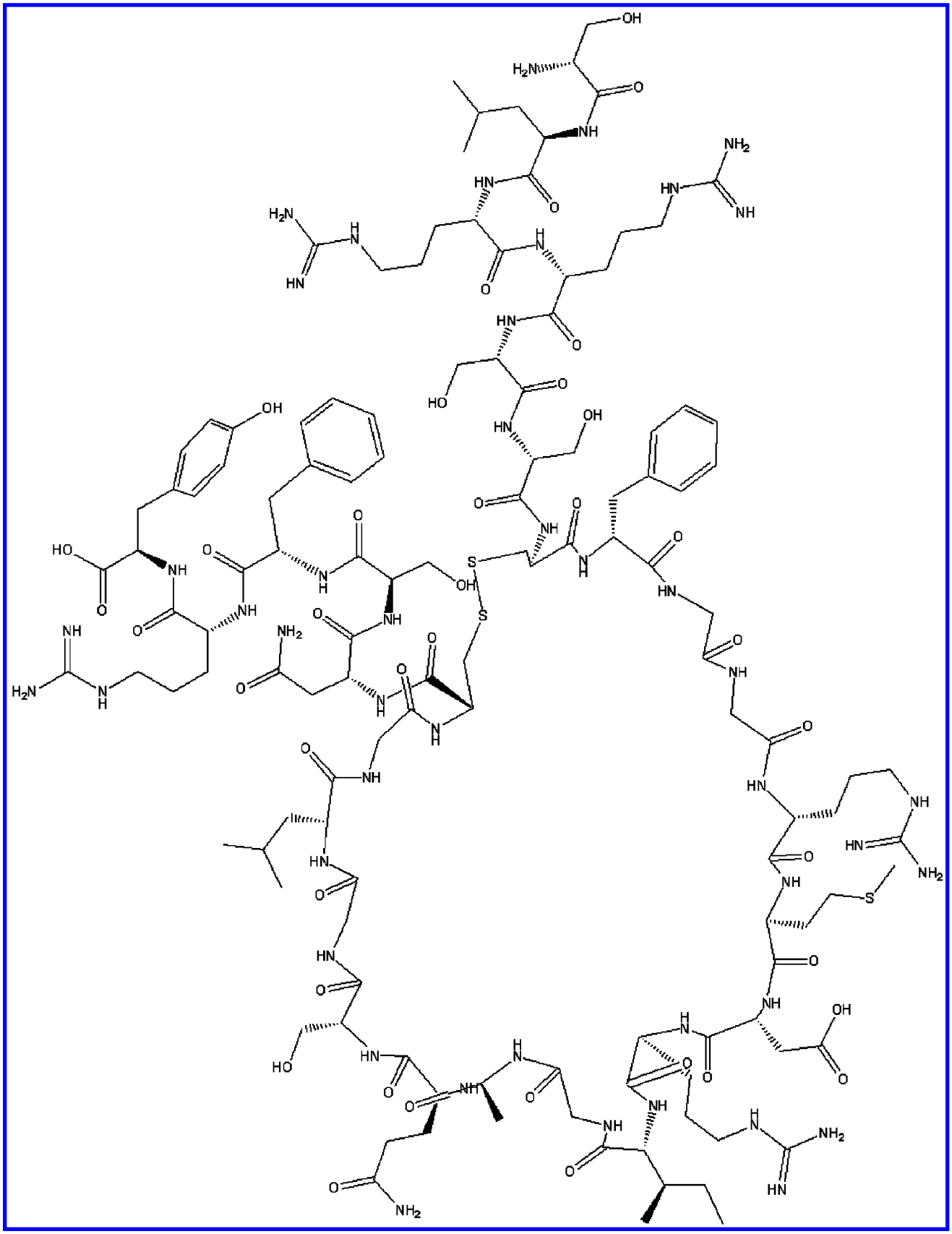

Figure 14. Structure of molecule **3904**.

small radii, which will produce a large number of singletons. Our experiments indicate that, for the Kazius data set, using radii beyond 5% of the maximum pairwise distance results in the detection of identical singletons, which correspond to the most isolated points in the descriptor space. At radii greater than 50% of the maximum pairwise distance, no singletons are detected. Though this discussion has focused on varying the LSH radius to explore chemical space, the low mean query times exhibited by the LSH algorithm make it an attractive tool to determine whether a new query point

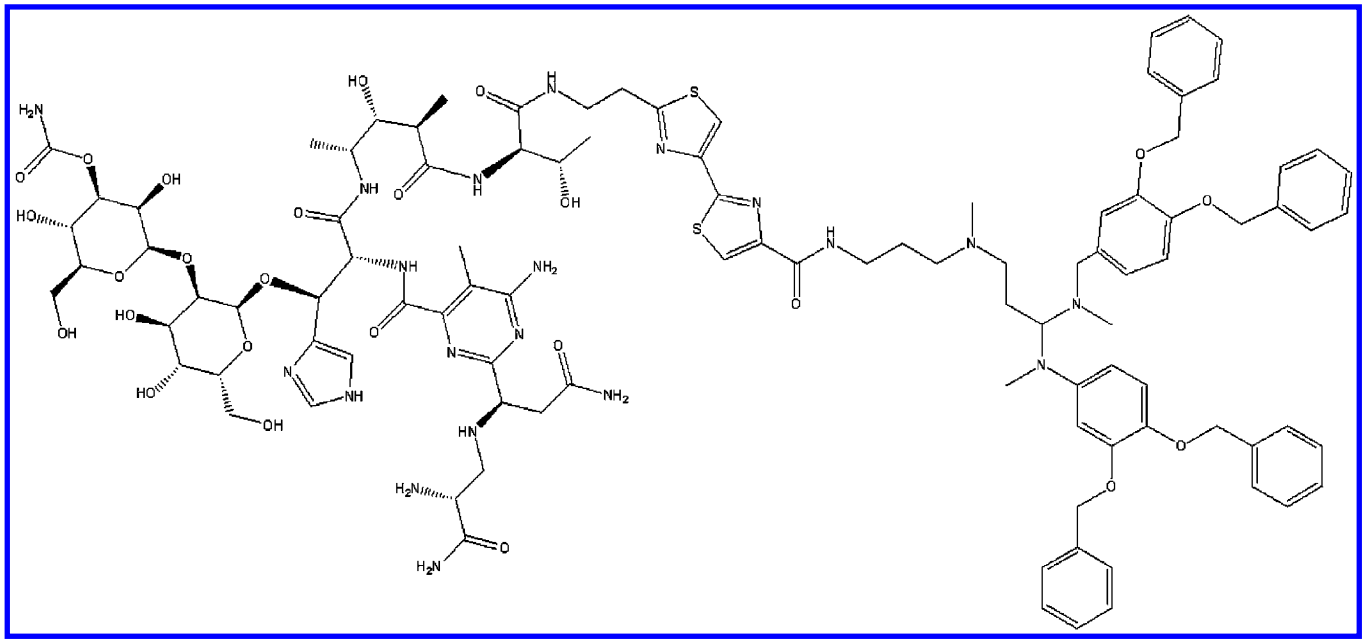

Figure 15. Structure of molecule **1861**.

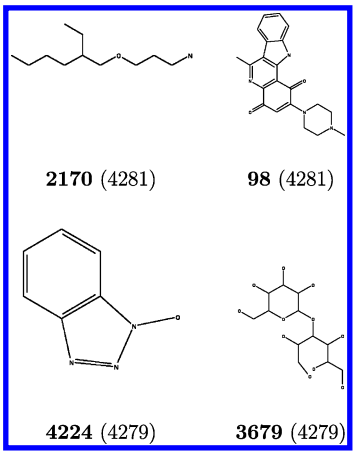

Figure 16. Some representative structures from the dense region of the Kazius data set, obtained by performing a LSH computation using 0.01% of the maximum pairwise distance. The values in parentheses indicate the number of nearest neighbors detected by the LSH algorithm at this radius.

lies in a dense region of the descriptor space or in a more sparse region of the space, given a precalculated LSH data structure.

4. FUTURE WORK

In this study, we have focused on a few fundamental features of the LSH algorithm that indicate its usefulness over traditional *k*NN methods. There are a number of other aspects of the LSH algorithm and its use that we are currently investigating.

One important aspect is the issue of data reduction and scaling. In this study, we only considered a simplistic approach to data reduction as a means of reducing the dimensionality of the descriptor space. Future work will involve a more extensive examination of the effects of different dimension reduction techniques on the LSH algorithm. Another aspect that was not considered in this study was the issue of scaling. In geometric terms, scaling the data set will shrink the extents of the descriptor space. This implies that the density of the space will increase. It would be expected that the increase in density of space would lead to increases in mean query times. Future work will examine how the density of a descriptor space affects the running time of the LSH algorithm.

In this study, we have only considered one sample application of outlier detection. To demonstrate the effectiveness of the method, future work will involve a more rigorous comparison of the LSH algorithm for outlier detection with other well-known outlier detection techniques as well as a more comprehensive application to other large data sets.

5. CONCLUSIONS

In this paper, we have presented a framework for mining large chemical libraries using some new algorithmic techniques in random projections. LSH uses random projections to hash nearby points to nearby bins. It guarantees sublinear time for approximate nearest-neighbor queries. We demonstrate the efficacy of this tool in fast outlier detection. The results indicate that the LSH algorithm can help speed up nearest-neighbor queries by 2 orders of magnitude compared to the naive *k*NN algorithm. We are not aware of any work that uses random projections followed by hashing to answer *k*NN queries in the cheminformatics literature. It should be noted that, for a few queries or a one-time analysis of a large combinatorial library, this approach does not offer significant advantages over the traditional *k*NN. However, if a library is to be repeatedly queried, the high speed exhibited by the LSH algorithm makes it an attractive approach. This would be especially applicable to chemical diversity problems where one would like to rapidly decide whether new compounds occupy a sparse region of a library's chemical space or not and, in either case, to determine which compounds they are similar to. In addition, even if a large library is analyzed once, any algorithm that makes multiple passes over the data (such as *k*-means) would benefit from an LSH-based preprocessing step. It is important to note that fingerprints can be used to rapidly determine which compounds in a library are similar to a query compound. However, this method, by definition, is restricted to binary fingerprints and

is effectively a linear scan. The use of the LSH algorithm allows one to perform such a search in sublinear time and also allows one to employ a variety of continuous descriptors to determine similarity. We believe that this is useful since spatial relationships are dependent on the descriptors used to define a chemical space. The LSH algorithm allows us to explore a variety of such spaces in a rapid fashion, which would not be feasible using the traditional *k*NN algorithm. As mentioned previously, another important use of the LSH algorithm would be to use it as a data partitioning scheme, whereby large libraries are divided into smaller chunks which can then be analyzed in detail by using classification or clustering techniques. This would be useful for the parallelization of the analysis of large chemical libraries.

The concept behind LSH is very simple, and the tool can be designed quite efficiently. Also, LSH requires minimal parameter tuning. The main parameter required is the error or approximation factor that we are willing to tolerate. In addition, LSH can be configured with a maximum limit on the available memory. Then, LSH can be made to self-tune the other parameters.

Because of the high accuracy of our LSH-based framework, as demonstrated in the previous sections, we can also conclude that LSH will be at least as accurate as any other standard *k*NN-based methods for classification. That is why we chose to investigate other data mining applications such as outlier detection. We have presented the LSH algorithm as a framework for working with large chemical data sets, and our results indicate that it provides an attractive alternative to traditional *k*NN algorithms in terms of time efficiency as well as flexibility.

ACKNOWLEDGMENT

We would like to thank Prof. P. Indyk and A. Andoni for providing us with the C++ source code for the LSH implementation.

REFERENCES AND NOTES

(1) Jorgensen, W. Design and Diversity Analysis of Large Combinatorial Libraries Using Cell-Based Methods. *Science* **2004**, *303*, 1813−1818.
(2) Datar, M.; Immorlica, N.; Indyk, P.; Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*; ACM Press: New York, 2004.
(3) Stahura, F.; Bajorath, J. Virtual screening methods that complement HTS. *Comb. Chem. High Throughput Screening* **2004**, *7*, 259−269.
(4) Xu, H.; Agrafiotis, D. Nearest Neighbor Search in General Metric Spaces Using a Tree Data Structure with a Simple Heuristic. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1933−1941.
(5) Cover, T.; Hart, P. Nearest Neighbor Pattern Classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21−7.
(6) Pearlman, R.; Smith, K. Metric Validation And The Receptor-Relevant Subspace Concept. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 28−35.
(7) Pearlman, R.; Smith, K. Novel Software Tools For Chemical Diversity. *Perspect. Drug Discovery Des.* **1998**, 339−353.
(8) Schnur, D. Design and Diversity Analysis of Large Combinatorial Libraries Using Cell-Based Methods. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 36−45.
(9) Gasteiger, J. *Chemoinformatics, A Textbook*; John Wiley & Sons: Weinheim, Germany, 2003.
(10) Duda, R.; Hart, P. *Pattern Classification*, 2nd ed.; Wiley-Interscience: Hoboken, NJ; 1998.
(11) Hastie, T.; Tibshirani, R.; Friedman, J. *An Introduction to Statistical Machine Learning*; Spinger Verlag: New York, 2001.
(12) Guha, S.; Meyerson, A.; Mishra, N.; Motwani, R.; O'Callaghan, L. Clustering Data Streams: Theory and Practice. *IEEE Trans. Knowledge Data Eng.* **2003**, *15*, 515−528.
(13) Gionis, A.; Indyk, P.; Motwani, R. Similarity Search in High Dimensions via Hashing. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, 1999.
(14) Agrafiotis, D. A Constant Time Algorithm for Estimating the Diversity of Large Chemical Libraries. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 159−167.
(15) Garey, R.; Johnson, D. *Computers and Intractibility*; W. H. Freeman: New York, 1979.
(16) Kazius, J.; McGuire, R.; Bursi, R. Derivation and Validation of Toxicophores for Mutagenicity Prediction. *J. Med. Chem.* **2005**, *48*, 312−320.
(17) Ames, B. N.; McCann, H.; Yamasaki, E. Methods for detecting carcinogens and mutagens with the Salmonella/mammalian-microsome mutagenicity test. *Mutat. Res.* **1975**, *31*, 347−364.
(18) http://cactus.nci.nih.gov/DownLoad/AID2DA99.sdz.
(19) Voigt, J.; Bienfait, B.; Wang, S.; Nicklaus, M. Comparison of the Open NCI Database with Seven Large Chemical Structural Databases. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 702−712.
(20) Molecular Operating Environment (MOE 2004.03); Chemical Computing Group Inc.: Montreal, Quebec, Canada.
(21) http://cactus.nci.nih.gov/DownLoad/NCI3DA99.sdz.
(22) *MACCS Fingerprints*; MDL Information Systems Inc.: San Leandro, CA.
(23) Wiener, H. Structural Determination of Paraffin Boiling Points. *J. Am. Chem. Soc.* **1947**, *69*, 17−20.
(24) Balaban, A. Higly discriminating distance based topological index. *Chem. Phys. Lett.* **1982**, *89*, 399−404.
(25) Kier, L.; Hall, L. *Molecular Connectivity in Chemistry and Drug Research*; Academic Press: New York, 1976.
(26) Kier, L. A shape index from molecular graphs. *Quant. Struct.-Act. Relat. Pharmacol., Chem. Biol.* **1985**, *4*, 109−116.
(27) Kier, L. Shape indexes for orders one and three from molecular graphs. *Quant. Struct.-Act. Relat. Pharmacol., Chem. Biol.* **1986**, *5*, 1−7.
(28) Kier, L. Distinguishing atom differences in a molecular graph index. *Quant. Struct.-Act. Relat. Pharmacol., Chem. Biol.* **1986**, *5*, 7−12.
(29) Kier, L.; Hall, L.; Murray, W. Molecular Connectivity I: Relationship to local anasthesia. *J. Pharm. Sci.* **1975**, *64*, 1971−1974.
(30) Kier, L.; Hall, L. *Molecular Connectivity in Structure Activity Analysis*; John Wiley & Sons: Hertfordshire, England, 1986.
(31) Kier, L.; Hall, L. Molecular Connectivity VII: Specific treatment to heteroatoms. *J. Pharm. Sci.* **1976**, *65*, 1806−1809.
(32) Gasteiger, J.; Marsili, M. A New Model for Calculating Atomic Charges in Molecules. *Tetrahedron Lett.* **1978**, *34*, 3181−3184.
(33) Gasteiger, J.; Marsili, M. Iterative partial equalization of orbital elektronegativity − a rapid access to atomic charges. *Tetrahedron* **1980**, *36*, 3219−3228.
(34) Ertl, P.; Rohde, B.; Selzer, P. Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment Based Contributions and Its Application to the Prediction of Drug Transport Properties. *J. Med. Chem.* **2000**, *43*, 3714−3717.
(35) *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2004; ISBN 3-900051-07-0.

CI050403O