Published in final edited form as:
J Chem Inf Model. 2012 April 23; 52(4): 891–900. doi:10.1021/ci200552r.

Speeding Up Chemical Searches Using the Inverted Index: the Convergence of Chemoinformatics and Text Search Methods

Ramzi Nasr[†], **Rares Vernica**[‡], **Chen Li**[¶], and **Pierre Baldi**[§,*]

[†]Department of Computer Science and Institute for Genomics and Bioinformatics, University of California, Irvine, Irvine, CA 92697-3435, USA

[‡]Department of Computer Science, University of California, Irvine, Irvine, CA 92697-3435, USA

[¶]Department of Computer Science and Institute for Genomics and Bioinformatics, University of California, Irvine, Irvine, CA 92697-3435, USA

[§]Departments of Computer Science and Biological Chemistry and Institute for Genomics and Bioinformatics, University of California, Irvine, Irvine, CA 92697-3435, USA

Abstract

In ligand-based screening, retrosynthesis, and other chemoinformatics applications, one of-ten seeks to search large databases of molecules in order to retrieve molecules that are similar to a given query. With the expanding size of molecular databases, the efficiency and scalability of data structures and algorithms for chemical searches are becoming increasingly important. Remarkably, both the chemoinformatics and information retrieval communities have converged on similar solutions whereby molecules or documents are represented by binary vectors, or fingerprints, indexing their substructures such as labeled paths for molecules and n-grams for text, with the same Jaccard-Tanimoto similarity measure. As a result, similarity search methods from one field can be adapted to the other. Here we adapt recent, state-of-the-art, inverted index methods from information retrieval to speed up similarity searches in chemoinformatics. Our results show a several-fold speed-up improvement over previous methods for both thresh-old searches and top-K searches. We also provide a mathematical analysis that allows one to predict the level of pruning achieved by the inverted index approach, and validate the quality of these predictions through simulation experiments. All results can be replicated using data freely downloadable from http://cdb.ics.uci.edu/.

Introduction

One of the most fundamental problems in chemoinformatics is to be able to search large databases of small molecules, such as PubChem[1] or ChemDB,[2] for molecules that are similar to a given query. Such searches are routinely performed in tasks ranging from ligand-based screening to retrosynthesis. Search efficiency is becoming increasingly important with the ongoing expansion of these databases due to the inclusion of newly synthesized molecules or even virtual molecules. A linear search comparing a query molecule to all the molecules in the databases one-by-one is often too slow and unnecessary. A number of data structures and algorithms have been developed over the years to speed up this process by pruning the search, i.e. by rapidly discarding molecules that are not similar to the query, without computing their similarity to the query.

[*]To whom correspondence should be addressed. pfbaldi@ics.uci.edu.

At a high level of conceptualization, the chemoinformatics similarity search problem appears to be analogous to the similarity search problem in information retrieval, for search engines and other applications, where a piece of text or a series of keywords are used to retrieve articles or web pages that are similar or pertinent to a given query. Upon close examination, however, this vague analogy becomes much more: an almost exact match, due to convergent evolution in the methods and data structures of two largely independent communities, the chemoinformatics community and the information retrieval community. This almost exact match suggests immediately that any procedure used in chemoinformatics can potentially be adapted to information retrieval, and vice versa. In particular, similarity searches in text retrieval are also carried using a pruning approach, but where the pruning is based on a particular data structure called the inverted index. As a result, it is natural to wonder whether the inverted index approach could be adapted to speed up chemical searches.

To investigate this question, here we first briefly review the representations and algorithms that are used for chemoinformatics searches. We then show how these are precisely connected to the representations and algorithms used in text retrieval and describe the inverted index approach. We then apply the inverted index approach to chemical searches and analyze its performance both analytically and through simulations.

Molecular Searches, Representations, and Similarity Scores

For clarity, it is useful to distinguish different kinds of database searches, which apply equally to databases of molecules, text, or other data. In *exact* searches, one is interested in finding out whether a database contains an entry identical to the query. Here we are primarily interested in *similarity* searches where one is interested in retrieving all the items in the database that are similar to the query, for some properly defined measure of similarity. A *threshold* similarity search retrieves all the items whose similarity to the query is above a given threshold, while a *top-K* similarity search retrieves the top K items most similar to the query. A threshold or top-K similarity search algorithm is *approximate* if it allows a small but non-zero false negative rate (i.e. some of the speed-up is obtained by missing a small fraction of the true positives[3]), otherwise it is said to be *exact*. Here we are primarily interested in exact threshold or top-K similarity search algorithms.

For databases of small molecules, several representations have been developed over the years, from one dimensional SMILES strings to 3D pharmacophores,[4] and different representations can be used for different purposes. To search large databases of molecules by similarity, most modern chemoinformatics systems use a binary fingerprint vector representation[4–10] whereby a molecule is represented by a vector whose components index the presence/absence of a particular feature, such as a functional group or substructure in the molecular graph. We use $\mathscr{A}$ to denote a molecule, $\vec{A} = (A_i)$ to denote the corresponding fingerprint of length $\vec{N}$, and A to denote the number of features, or 1-bits, present in the fingerprint $\vec{A}$ ($A = |\vec{A}|$).

In early chemoinformatics systems, fingerprint vectors were relatively short, containing typically a few dozen components selected from a small set of features, hand-picked by chemists. In most modern systems, however, the major trend is towards the combinatorial construction of extremely long feature vectors with a number of components N that can vary in the $10^3 – 10^6$ range, depending on the set of features. Examples of such combinatorial features include all possible labeled paths or labeled trees (circular substructures) up to a certain length or depth. In many chemoinformatics systems, the resulting long and sparse fingerprint vectors are further compressed to much shorter and denser binary fingerprint vectors. A widely used method of compression is a lossy compression method based on the application of the logical OR operator to the binary fingerprint vector after modulo wrapping to 512, 1,024, or 2,048 bits.[7] Other more efficient loss-less methods of compression have recently been developed.[10] The inverted index approach to be described and the corresponding analytical results apply to all the different kinds of binary finger-prints. Extension of the method to non-binary fingerprints, such as fingerprints based on counts or weighted fingerprints, is possible and a matter of ongoing research that will not be discussed here. The particular fingerprints and features used in our experiments are described in the Empirical Results section.

Several similarity measures have been developed for molecular fingerprints.[11,12] The most widely used similarity measure for chemical database searches is the Tanimoto-Jaccard (or just Tanimoto) similarity. Given two molecules $\mathscr{A}$ and $\mathscr{B}$, the Tanimoto similarity is given by

$$S(\mathscr{A},\mathscr{B})=S(\vec{A},\vec{B})=\frac{A\cap B}{A\cup B}=\frac{A\cap B}{A+B-A\cap B} \quad (1)$$

Here $A \cap B$ denotes the size of the intersection, i.e. the number of features present in both $\vec{A}$ and $\vec{B}$, and $A \cup B$ denotes the size of the union, *i.e.* the number of features present in $\vec{A}$ or $\vec{B}$. In the remainder of this article we focus on the Tanimoto similarity, although it should be clear that the same ideas apply immediately to the other similarity measures described in[11] since these are based on the same intersection and union ingredients, which are at the root of the inverted index approach.

Previous Chemoinformatics Work

To improve search speed, previous methods exploit various data structures and bounds on the Tanimoto similarity to prune the search space. For clarity, the data structures can be separated into hashing and tree data structures. The essence of the pruning is to note that any bound on the intersection of two fingerprints derived from the data structures, results immediately into a bound on their Tanimoto similarity, since whenever we have $A \cap B < T$ then $S(\vec{A},\vec{B}) < T/(A + B - T)$. When favorable, the bounds eliminate database molecules that are guaranteed not to be in the desired list, thereby reducing the number of pair-wise similarity computations between database molecules and the query. Thus in these methods the pruning is done "by molecules" (rows), whereas in the inverted index approach we are about to present the pruning is done "by features" (columns).

Hashing Data Structures

The basic idea in this class of methods is to pre-compute for each fingerprint in the database a short signature which can be used to rapidly bound the Tanimoto similarity. This is done by first partitioning the components of the fingerprints into M ($1 \le M \le N$) fixed sets and then using a simple function f to summarize the complement of 1-bits that are observed in each set for each molecule. While the M sets do not need to be of the same size and could be chosen according to some specific strategy, in most cases one uses sets of equal size and randomly selected, which is equivalent to randomly permuting the fingerprint components and putting all the components from the same class modulo M in the same set. For instance, if $M = 2$ all the even components are in one set, and all the odd components are in the other set. During a search, the query fingerprint's signature is constructed as determined by M and f, and bounds on the Tanimoto similarity are calculated as a function of the query signature and the pre-computed database signatures.

Examples of specific implementations in this class include:

1. When f is the logical *OR* operator capturing whether there is at least one 1-bit in a given set (with, for instance, $M = 1{,}024$), one obtains the lossy OR-compressed

fingerprint used in the Daylight system[7] and the Tanimoto similarity computed on these compressed fingerprints can be used to search the database directly, although even better results can be obtained with a systematic correction.[13]

2. When f is the logical *XOR* operator capturing whether the total number of 1-bits in a given partition is odd or even (with, for instance, $M = 128$), one obtains the approach described in Baldi et al.[14] and the corresponding bounds to prune the database.

3. When f is the sum operator, the signature corresponds to the total number of bits equal to 1 in each of the M sets, and one then can use the bounds[15–19] on the Tanimoto similarity to prune the database. When $M = 1$, the signature simply represents the total number of 1-bits in the corresponding fingerprint. We refer to this special case as the *BitBound* method. The case $M = 1$ and the general case are studied in Swamidass et. al[15] and Nasr et al.[19] respectively.

Tree Data Structures

The problem of exact search in a database of binary vectors is typically addressed by organizing the database into a binary tree, with splits that are roughly even at each step to maximize the entropy of the corresponding decision. The same idea can be applied to similarity search where it is called the *MultiBit* Tree method.[20] In the pre-processing steps, starting with all the fingerprints assigned to the root node, the algorithm recursively partitions the fingerprints into two sets. Each fingerprint is assigned exclusively to one of the child nodes according to the following criterion. For each fingerprint $\vec{B}$ in the parent node, the algorithm inspects a feature position s and assigns $\vec{B}$ to one child node if $B_s = 1$, and to the other child if $B_s = 0$. The feature position s is chosen by maximum entropy to keep the tree as balanced as possible i.e. there should be roughly as many fingerprints with $B_s = 1$ as those with $B_s = 0$. Along any branch, the splitting process stops whenever the number of fingerprints in a node falls below a preset level. After each split, the algorithm records the list of all components that have constant value 0 or 1 among all the remaining fingerprints. Specifically, after each split, the algorithm records the list $\mathcal{O} = (o_1, o_2, \ldots, o_j, \ldots o_{|\mathcal{O}|})$ of all the columns where all the remaining fingerprints have all bit positions equal to 0 ($B_{o_j} = 0$), and the list $\mathcal{I} = (i_1, i_2, \ldots, i_k, \ldots, i_{|\mathcal{I}|})$ of all the columns where all the remaining fingerprints have all bit positions equal to 1 ($B_{i_k} = 1$). By definition, the splitting bit position s belongs to $\mathcal{O}$ for one child node and to $\mathcal{I}$ for the other child node. During a search, the algorithm visits the nodes in depth-first traversal. At each node, a bound on the Tanimoto similarity is computed as a function of $\mathcal{O}$ $\mathcal{I}$, and the query fingerprint. This bound becomes tighter with the depth of each branch, allowing pruning entire subtrees of molecules from the search when the bound becomes favorable.

Convergent Evolution Between Chemoinformatics and Information Retrieval

In information retrieval, one is typically interested in searching a database of text documents (e.g. articles, web pages, strings) for documents that are related to a query string, consisting for instance of keywords. While molecules are sometimes represented by strings, in the form of SMILES strings, it is even more remarkable that documents are most of the time represented by fingerprints. When the features are the words of the dictionary, one obtains the well-known “bag of words” representation, which discards any sequential information. Alternatively, when the documents are viewed as strings, one can use n-grams–sequences of n symbols from the alphabet (which can include space, punctuation marks, and other symbols)–as the dictionary of features. An English dictionary of relatively common words can have $10^4 – 10^6$ words, and all possible n-grams over an alphabet of 50 characters when n

varies from 3 to 4 cover a range of 125×10^3 to $6{,}250 \times 10^3$. Thus the ranges of N for molecule and document fingerprints are comparable. Finally, the similarity measure of choice in many information retrieval systems is also the Jaccard-Tanimoto similarity measure. There are of course some differences between the two fields. For example, although the size of the databases are generally also comparable, document databases can be larger than the databases of commercially available compounds since, for instance, the total number of existing web pages is in the range of many billions whereas the total number of commercially available compounds is at most a few tens of millions. Similarly, hyperlinks or other referential connections between documents are well developed and used in the well-known PageRank algorithm in information retrieval, whereas a system of connections between molecules is less used or well-established. Nevertheless, it is remarkable that the two fields have independently converged onto identical representations and similarity measures.

The inverted index is a data structure routinely used in information retrieval which inverts the fingerprint vector representation by associating each feature with the list of documents in the database containing that feature[21,22] (Figure 1). The notion of an inverted index, or inverted file, or inverted search is natural and of course not new. It has been used in computer science[23] and other sciences, including chemistry,[24–26] for several decades. The inverted index can be constructed for molecular fingerprints as easily as for documents. However, past applications of the inverted index to chemoinformatics (for example, see[27]), have been very simple, and have not used the most sophisticated methods that have been developed for text data and search engines in information retrieval. [As a side note, the inverted index approach for text data can also be applied directly to SMILES string representations for molecules. Kristensen[28] did so after being introduced to the inverted index in our laboratory]. Here we focus on applying the state-of-the-art inverted index algorithms from information retrieval to speed up fingerprint similarity searches.

Using the Inverted Index for Similarity Search

In a similarity threshold search with threshold t, given a query molecule $\mathcal{A}$ one is interested in finding all the molecules $\mathcal{B}$ in the database such that

$$S(\vec{A},\vec{B})=\frac{A \cap B}{A \cup B}>t \quad (2)$$

Given the values of A and B, the bound on the similarity is equivalent to a bound on the intersection

$$S(\vec{A},\vec{B})=\frac{A \cap B}{A \cup B}=\frac{A \cap B}{A+B-A \cap B}>t \iff A \cap B>T=\frac{t\,(A+B)}{t+1} \quad (3)$$

Thus, given the values of A and B, the similarity threshold problem is equivalent to the T occurrence problem, finding all fingerprints with an intersection of size at least T with the fingerprint query, to which the inverted index can be applied.[29]

In our setup, we first perform a preprocessing step such that all molecules $\mathcal{B}$ are binned according to the fingerprint size B.[15,19] During a search, we first prune the search space by discarding bins corresponding to $B \leq tA$ or $B \geq A/t$. This first pruning step is a result of the *BitBound* method[15] introduced above. Within each of the remaining bins ($tA < B < A/t$), we then find molecules, $\mathcal{B}$, that satisfy Equation Eq. (3) using an inverted index data structure for each bin of fingerprint sizes. Thus the inverted index feature lists are created within each bin during the preprocessing steps.

During a search, within the bins that pass the first pruning step, only the feature lists that correspond to the features present in the query fingerprint are kept. For each bin, we calculate T from Equation Eq. (3) using three fixed parameters: B is fixed within the bin, t is the given Tanimoto threshold, and A is the query fingerprint size. The direct advantage of binning is two-fold: (1) the first pruning step narrows the search space; and (2) fixing B within each bin allows the calculation of T. Several algorithms have been developed to compute the solution to the T-occurrence problem efficiently for text data.[29] Here we focus on *DivideSkip*, the algorithm that performs the best on the chemical fingerprint data.

Inverted index algorithm

As part of our pre-processing steps, we sort the molecular identifiers in each feature list (Figure 1) and order the feature lists by their sizes (the number of identifiers they contain) (Figure 2). Only the A features associated with 1-bits in the query are retained. The number A of retained features must be greater or equal to T otherwise no molecule can have an intersection of size at least T with the query. The algorithm then splits the feature lists into 2 sets: a set of L ($0 \leq L \leq T$) longest lists ($£_{Long}$), and a set ($£_{Short}$) containing the remaining $A - L$ shorter lists. The two sets are created to take advantage of efficient search algorithms whose efficiency depends on the list sizes. Note that a molecule can occur at most L times in $£_{Long}$. Therefore a molecule with an intersection with the query greater than T must occur at least $T - L$ times within $£_{Short}$. Thus in the next phase the algorithm first searches for molecular identifiers that occur at least $T - L$ times in $£_{Short}$, and then uses these identifiers to search $£_{Long}$. More specifically, the algorithm uses a Heap data structure to search for molecular identifiers that occur at least $T - L$ times in $£_{Short}$ and for each one of them counts the number of its occurrences. The Heap keeps track of the top elements of the sorted feature lists and efficiently skips elements that are guaranteed not to occur at least $T - L$ times (Figure 3). The resulting molecules and their counts are then used to search the long lists in $£_{Long}$ using binary search and their counts are further updated. Only those with counts greater or equal to T are retained. The parameter L discussed in the Results must be chosen to optimize the tradeoffs between the Heap algorithm of $£_{Short}$ and the binary search algorithm of $£_{Long}$.

In the case of top-K searches, T cannot be fixed as in threshold searches because it depends on the query and K. The algorithm we use in this case starts with $T = 0$ occurrences, and updates T as it fills a sorted array of size K. This one-pass algorithm returns the K most similar results using a variant of *DivideSkip*. Detailed descriptions and evaluations of the top-K algorithm and *DivideSkip* can be found in[29,30] for text data.

Theoretical Results: Pruning Rate Prediction

The approach we have described has three pruning steps and in order to analytically predict the overall pruning, one needs to predict the amount of pruning associated with each step. The first pruning step is a direct consequence of binning fingerprints and discarding bins corresponding to $B \leq tA$ or $B \geq A/t$ as described in previous work.[15,19] Analytical results that predict the pruning at the first pruning step, given the mean and standard deviation of B and the threshold t, are described in.[15,19] The second pruning step comes from retaining only molecules associated with features that are present in the query molecule. For fingerprints based on a large basis of combinatorial features, such as the labeled paths or trees considered here, the amount of pruning coming from this step is negligible. This is because some basic features (e.g. a path containing only carbon atoms) are found in almost all molecules and therefore are not discriminative. Thus here we do not consider this pruning step any further, although it could become significant when other basis of features are used, in which case the amount of pruning could be predicted using the same techniques used below to estimate the amount of pruning associated with the third step. The third pruning step is the most novel and relevant for this paper and is a direct consequence of the inverted index *DivideSkip* algorithm described above. We have seen that when searching for molecules with at least T occurrences in the ranked feature lists associated with the query, the algorithm splits these lists into two subsets $\pounds_{Short}$ and $\pounds_{Long}$. Any molecule that does not occur at least $T - L$ times in $\pounds_{Short}$ is pruned from the search. Thus to estimate the amount of pruning resulting from the third step we need to estimate how many molecules occur less than $T - L$ times in $\pounds_{Short}$, as a function of A, B, T, L, and the size D of the database. The analysis proceeds in four steps:

1. Choosing a probabilistic model of fingerprints.
2. Selecting query feature lists using the probabilistic model.
3. Approximating the distribution of occurrences in the selected feature lists.
4. Deriving the probability of pruning.

Choosing a probabilistic model of fingerprints

The simplest probabilistic model of fingerprints is a sequence of N independent identically distributed Bernoulli trials (coin flips) with probability p of producing a 1-bit.[14,15,19,31] This corresponds to a Binomial model $\mathcal{B}(N, p)$, with only two parameters N and p, where N is the number of possible features in a fingerprint. The bits in this model are statistically exchangeable and independent. While such a model can be useful to derive a number of approximations, fingerprint features are not exchangeable since the individual bit probabilities $p_1,\ldots, p_N$ are not identical and equal to p, but vary significantly.[10] The strong exchangeability assumption of the Binomial model directly affects the sizes of the feature lists. Using a single probability p for all bits "flattens" the individual bit probabilities, thereby resulting in exchangeable equally-sized feature lists. As a consequence, the Binomial model does not produce a realistic approximation of the long and short feature lists of the *DivideSkip* algorithm.

The statistical model at the next level of approximation is that of a sequence of non-stationary independent coin flips where the probability p_i of each coin flip varies. This Multiple-Parameter Bernoulli model has N parameters: $p_1, p_2, \ldots, p_i, \ldots, p_N$. These parameters can be obtained from the entire database, or from the fingerprints that have B 1-bits when dealing with the corresponding bin. The latter case is expected to give more accurate results at the expense of having multiple sets of parameters, one for each B. Figure 4 shows how the N parameters vary with different values of B for the fingerprints described in the Empirical Results section. Furthermore, query and database fingerprints can be modeled with different bit probabilities. To distinguish the two models in the derivations, we use $\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_i, \ldots, \bar{p}_N$ to denote the parameters of the Multiple-Parameter Bernoulli model for the query. Similarly, the $\bar{p}_i$ parameters can be conditioned on A, the number of 1-bits in the query. We choose to use the Multiple-Parameter Bernoulli model as it provides a better approximation of the long and short feature lists. Although the Multiple-Parameter Bernoulli model ignores the weak correlations between the fingerprint components, it has been shown in previous publications to perform quite well in a variety of calculations.[20,31]

Selecting feature lists using the probabilistic model

During a search, only the feature lists that correspond to features present in the query fingerprint are selected as input to the *DivideSkip* algorithm. Hence, if the query fingerprint $\vec{A}$ has A 1-bits, A feature lists are selected, and one can continue the analysis with these features if the goal is to estimate the pruning associated with this particular query. However, it is more interesting to estimate the average amount of pruning across many queries. Integrating over all possible queries is not tractable analytically. Thus here we seek to obtain an estimate of the expectation by looking at what happens in the case of a typical molecule. To model the A feature lists that are used in the *DivideSkip* algorithm, we need to reduce the N probabilities, $\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_i, \ldots, \bar{p}_N$ characterizing the typical query to A typical probabilities, $r_1, r_2, \ldots, r_j, \ldots, r_A$ that are representative of the selected feature lists. This can be done with the following two-step approach.

Without any loss of generality, assume that the probabilities $\bar{p}_i$ are arranged in increasing order (as shown in Figure 4). A first step is to divide the sorted probabilities, $\bar{p}_i$, into A equally sized and consecutive sets, and calculate r_j as the mean of the $\bar{p}_i$'s in each set. However, having equally sized sets ignores the non-uniformity of bit probabilities in the query fingerprint: $\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_i, \ldots, \bar{p}_N$. To demonstrate this point, consider for example the A^{th} set which contains the highest N/A probabilities of the Multiple Parameter Bernoulli model. A typical query molecule contains multiple features that correspond to the high probabilities in this set, yet this approach under-represents them with one mean value. Conversely, the equally sized sets over-represent the low probabilities for low values of j. Thus a better approach, corroborated by our experiments, is to use a second subsequent step to divide the $\bar{p}_i$ probabilities into A sets, where for every j the size of set j is inversely proportional to the value of r_j computed in the first step, and then re-estimate each r_j by the mean of the $\bar{p}_i$'s in the corresponding set. Features corresponding to high values of $\bar{p}_i$ occur frequently in query fingerprints. As a result, this second step groups high $\bar{p}_i$'s into smaller sets, associating a larger number of such small, large-average-probability, sets with the r_j's. Likewise, this approach associates a smaller number of large, small-average-probability, sets with the r_j's. In the simulations, for simplicity we assume that the query is generated from the same probabilistic model as the entire databases. Thus we use this two-step approach with $\bar{p}_i = p_i$ ($i = 1,\ldots,N$) to generate the values of r_i ($i = 1,\ldots, A$).

Approximating the distribution of occurrences in the selected feature lists

In the *DivideSkip* algorithm, the A selected feature lists are split into $£_{Long}$ (L longest lists) and $£_{Short}$ (remaining $A - L$ lists). Molecules occurring less than $T -$ times in $£_{Short}$ are pruned. In order to predict the pruning, we need to approximate the distribution of the number of occurrences of a molecule in $£_{Short}$. Let M denote the number of occurrences of a molecule in $£_{Short}$. The probabilities of the $£_{Short}$ lists ($r_1, r_2,\ldots, r_j,\ldots, r_{A-L}$) can be used to derive the expected value of M. Assuming independence between the feature lists, the expectation of M is given by

$$E[M] = r_1 + r_2 + \ldots + r_{A-L} = \sum_{j=1}^{A-L} r_j \quad (4)$$

and the variance by

$$\mathrm{Var}[M] = r_1(1 - r_1) + \ldots + r_{A-L}(1 - r_{A-L}) = \sum_{j=1}^{A-L} r_j(1 - r_j) \quad (5)$$

The distribution of M can be approximated by a Gaussian distribution with parameters $E[M]$ and $\mathrm{Var}[M]$ or, even better in the regime of rare events typically associated with the sparse columns of $£_{Short}$, by a Poisson distribution with $\lambda = E[M] = \mathrm{Var}[M]$. Using the Poisson approximation gives: $P(M = k) = e^{-\lambda} \lambda^k / k!$

Deriving the probability of pruning

The probability of pruning a molecule at a given T and L is given by the cumulative probability of obtaining less than $T - L$ occurrences:

$$P\ (M<T\ -\ L)=\sum_{k=1}^{T-L-1}P(M=k)=\sum_{k=1}^{T-L-1}e^{-\lambda}\frac{\lambda^k}{k!}\quad (6)$$

the last equality resulting from the Poisson approximation. The expected total number of molecules being pruned is thus $DP(M < T - L)$ or $D_B P(M < T - L)$ depending on whether one is applying the analysis to the entire database of D fingerprints, or the bin containing D_B fingerprints of size B.

Empirical Results

Implementation and experimental setup

In the simulations, for completeness we use both fingerprints based on labeled paths up to depth 8, and fingerprints based on labeled circular substructures of depth up to 2, with Extended Connectivity (EC) labeling.[32,33] The paths and EC features are among the most used in chemoinformatics databases. We conduct simulations with both uncompressed fingerprints[10] and with compressed fingerprints using the lossy OR compression modulo 1024 (similar to the Daylight fingerprints[7]). Table 1 shows the empirical mean and standard deviation of the number of 1-bits for different kinds of fingerprints.

In all experiments, *DivideSkip* is run in comparison to the *Hashing* method[19] and the *MultiBit Tree* method[20] from previous work. The value of L that optimizes the various tradeoffs and results in close-to-minimal search time can be estimated as a function of T and two other parameters:

$$L=\frac{T}{\mu\ \log\ M+1}\quad (7)$$

where M is the size of the longest feature list and μ is a query independent and data set dependent coefficient.[29] As described in the previous section, the amount of pruning can also be predicted analytically as a function of L without the need for multiple empirical runs. The *Hashing* method has a single parameter, the hash size M, which is set at 128 in the experiments.[19] The *MultiBit Tree* method also has a single parameter, the maximal number of fingerprint associated with a leaf, which is set to 10 fingerprints.[20] The parameter values above result from a parameter sweep to identify values that optimize speed. All the search methods and the testing framework are implemented in the C/C++ language.

The timing experiments are obtained by running 100 threshold and top-K search queries against a sample of 100,000 database molecules. The query and database molecules are randomly sampled from ChemDB. The time of each threshold search is recorded at multiple Tanimoto thresholds between 0 and 1, and of each top-K search at multiple typical values of K: [1,5,10,20,50,100]. All the experiments are carried out on a Intel® Xeon™ 3.00 GHz processor with 1 Gbytes of RAM.

Pruning results

Simulations show that the analytical formula for predicting the amount of pruning of the inverted index algorithm (Eq. (6)) is quite accurate, over a wide range of parameters. For instance, Figure 5 shows the predicted (dotted line) and empirical (solid line) pruning rates at multiple values of L, for a typical bin with $B = 200$ and a typical query with $A = 200$, for $T = 100$ and $T = 140$. In general, the predicted values deviate by less than 0.1 from the empirical values over a wide range of L.

Timing results

For the timing experiments, we report the mean and standard deviation for each of the 100 search queries against the 100,000 ChemDB molecules. Showing the results for 100 queries proves sufficient as the mean and standard deviations are similar for larger query sets. Figure 6 shows the results of the threshold search. The *DivideSkip* method outperforms the previous approaches in all fingerprint cases: with paths and tree features, and with or without lossy compression. It performs up to 8 times faster than the *Hashing* method in the case of EC labeling on uncompressed fingerprints (top right panel of Figure 6) and up to 3 times faster than the *MultiBit Tree* in paths labeling on uncompressed fingerprints (bottom right panel of Figure 6). The performance of *DivideSkip* is only 1.8 times better than that of the *Hashing* method in the case of paths labeling on uncompressed fingerprints (bottom right panel of Figure 6), and only 1.3 times better than that of the *MultiBit Tree* in paths labeling on compressed fingerprints (bottom left panel of Figure 6). As can be expected, the differences in timing become small as the threshold approaches one, since most molecules in the database are pruned from the search in all the approaches. More importantly, the results of the top-K search are shown in Figure 7. The inverted index method once again outperforms the previous approaches in all fingerprint cases. It performs up to 5 times faster than the *Hashing* approach and up to 4 times faster than the *MultiBit Tree* approach in the case of EC labeling on uncompressed fingerprints (top right panel of Figure 7), and between 2 to 3 times faster than the best alternative in all other cases.

Table 1 demonstrates the effect of the feature basis and compression algorithm on the number of features per fingerprint. As expected, lossy compressed fingerprints contain less features than uncompressed fingerprints. Also, there are more features per fingerprints with paths labeling than with EC labeling. As a consequence, there are more and longer feature lists in the input when using paths labeling versus EC labeling, and more feature lists when using uncompressed fingerprints versus compressed fingerprints. This is reflected in the timing results of Figure 6 and Figure 7. In both figures the performance of *DivideSkip* on EC-labeled compressed fingerprints and EC-labeled uncompressed fingerprints are very similar (top left and top right panels), as is the corresponding number of features per fingerprints in Table 1. Note that uncompressed fingerprints with EC labeling lose less information under lossy compression than paths because they do not contain as many features per fingerprint.

Discussion and Conclusion

The chemoinformatics and information retrieval communities have worked for decades on the same basic problem, similarity search of large datasets, but with different data: molecules in one case, text in the other. Remarkably, these two communities have converged independently onto the same binary fingerprint representations, and the same Tanimoto-Jaccard similarity measure. Thus methods developed by one community may be applicable to the other, as shown in this paper by applying an inverted index approach originally developed for text data to chemical data. Theoretical analyses and simulations presented here establish the efficiency of the inverted index algorithm for chemical fingerprint data.

The parameters of the fingerprint representations and databases vary in each field and each application but by and large remain in comparable ranges. For example, Table 2 compares the number of distinct grams of size 3 empirically observed in various text data sets[29] to the number of paths and EC features empirically observed in the ChemDB dataset used here. Note that these numbers, which depend on the size of the data set, are much lower than the total number of possible grams and chemical features (it is estimated that the number of possible EC labels with depth up to 3 exceeds four billion (4×10^9)[32,33]).

Another important parameter that turns out to be comparable in many cases is the typical number of 1-bits in the fingerprints. The number of grams per string is a direct consequence of the string's length as grams are obtained by sliding a window over the string. In text searches, strings are typically small which results in a small number of grams per string. For example, the average size of strings is ~ 20 characters in the case of proper names in the IMDB data set,[29] or slightly higher in other data sets such as addresses or paper titles, which indicates that the number of grams per string is also small. For chemical fingerprints, the number of features that occur in a molecule is also small (and only weakly coupled to the size of the molecule), resulting also in sparse fingerprint vectors.

Finally, the distribution of the list sizes is also an important consideration for the inverted index approach. For both text and chemical data, it is well known that the distribution of the feature frequencies follows approximately a power-law distribution.[34,35] Thus the parameters and distributions similarities between chemical and text data explain in part why the inverted index approach works so well in two fields that are so different. The approach is also flexible with respect to the growth of a given database. As long as the feature basis remains the same, when a new molecule is added to the database the inverted index can be updated easily by appending the new molecule to the end of the corresponding feature lists. Thus, while many other considerations may enter into the implementation of a specific chemoinformatics database, the inverted index data structure and algorithmic approach provide considerable speed and flexibility and ought to be considered as prime candidates for the similarity search component of the implementation.

Acknowledgments

Work supported by NIH grants LM010235-01A1 and 5T15LM007743, NSF grant 0513376, and a Microsoft Research Award to PB. We would like to acknowledge the OpenBabel project, and OpenEye Scientific Software for their free software academic licenses.

References

1. Wang Q, Ng D, Mannan MS. Study on the Reaction Mechanism and Kinetics of the Thermal Decomposition of Nitroethane. Ind. Eng. Chem. Res. 2009; 48:8745–8751.
2. Chen J, Swamidass SJ, Dou Y, Bruand J, Baldi P. ChemDB: a Public Database of Small Molecules and Related Chemoinformatics Resources. Bioinformatics. 2005; 21:4133–4139. [PubMed: 16174682]
3. Tabei Y, Tsuda K. SketchSort: Fast All Pairs Similarity Search for Large Databases of Molecular Fingerprints. Mol. Inf. 2011; 30:801–807.
4. Leach, AR.; Gillet, VJ. An Introduction to Chemoinformatics. Berlin: Springer; 2005.
5. Fligner MA, Verducci JS, Blower PE. A Modification of the Jaccard/Tanimoto Similarity Index for Diverse Selection of Chemical Compounds Using Binary Strings. Technometrics. 2002; 44:110–119.
6. Flower DR. On the Properties of Bit String-Based Measures of Chemical Similarity. J. Chem. Inf. Comput. Sci. 1998; 38:379–386.
7. James CA, Weininger D, Delany J. Daylight Theory Manual. 2004 Available at http://www.daylight.com/dayhtml/doc/theory/index.html.
8. Xue L, Godden JF, Stahura FL, Bajorath J. Profile Scaling Increases the Similarity Search Performance of Molecular Fingerprints Containing Numerical Descriptors and Structural Keys. J. Chem. Inf. Comput. Sci. 2003; 43:1218–1225. [PubMed: 12870914]
9. Xue L, Stahura FL, Bajorath J. Similarity Search Profiling Reveals Effects of Fingerprint Scaling in Virtual Screening. J. Chem. Inf. Comput. Sci. 2004; 44:2032–2039. [PubMed: 15554672]
10. Baldi P, Benz RW, Hirschberg D, Swamidass S. Lossless Compression of Chemical Fingerprints Using Integer Entropy Codes Improves Storage and Retrieval. J. Chem. Inf. Model. 2007; 47:2098–2109. [PubMed: 17967006]

11. Holliday JD, Hu CY, Willett P. Grouping of Coefficients for the Calculation of Inter-Molecular Similarity and Dissimilarity Using 2D Fragment Bit-Strings. Comb. Chem. High. Throughput Screen. 2002; 5:155–166. [PubMed: 11966424]

12. Nasr R, Swamidass SJ, Baldi P. Large Scale Study of Multiple-Molecule Queries. J. Cheminf. 2009; 1:7.

13. Swamidass SJ, Baldi P. Mathematical Correction for Fingerprint Similarity Measures to Improve Chemical Retrieval. J. Chem. Inf. Model. 2007; 47:952–964. [PubMed: 17444629]

14. Baldi P, Hirschberg DS, Nasr RJ. Speeding up Chemical Database Searches Using a Proximity Filter Based on the Logical Exclusive OR. J. Chem. Inf. Model. 2008; 48:1367–1378. [PubMed: 18593143]

15. Swamidass SJ, Baldi P. Bounds and Algorithms for Fast Exact Searches of Chemical Fingerprints in Linear and Sublinear Time. J. Chem. Inf. Model. 2007; 47:302–317. [PubMed: 17326616]

16. Burkhard W, Keller R. Some Approaches to Best-Match File Searching. Commun. ACM. 1973; 16:230–236.

17. Shapiro M. The choice of Reference Points in Best-Match File Searching. Commun. ACM. 1977; 20:339–343.

18. Baldi P, Hirschberg DS. An Intersection Inequality Sharper than the Tanimoto Triangle Inequality for Efficiently Searching Large Databases. J. Chem. Inf. Model. 2009; 49:1866–1870. [PubMed: 19601605]

19. Nasr R, Hirschberg DS, Baldi P. Hashing Algorithms and Data Structures for Rapid Searches of Fingerprint Vectors. J. Chem. Inf. Model. 2010; 50:1358–1368. [PubMed: 20681581]

20. Nasr R, Kristensen T, Baldi P. Tree and Hashing Data Structures to Speed up Chemical Searches: Analysis and Experiments. Mol. Inf. 2011; 30:791–800.

21. Zobel J, Moffat A. Inverted Files for Text Search Engines. ACM Computing Surveys (CSUR). 2006; 38:6.

22. Manning, C.; Raghavan, P.; Schutze, H. Introduction to Information Retrieval. Vol. Vol. 1. Cambridge, UK: Cambridge University Press; 2008.

23. King D. The Binary Vector as the Basis of an Inverted Index File. Journal of Library Automation. 1974; 7:307–314.

24. Matthews F. Searching X-Ray Diffraction Powder Data with an Inverted Coordinate Index. J. Chem. Doc. 1963; 3:213–216.

25. Thomson LH, Hyde E, Matthews FW. Organic Search and Display using a Connectivity Matrix Derived from Wiswesser Notation. J. Chem. Doc. 1967; 7:204–209.

26. Hoffman WS. An Integrated Chemical Structure Storage and Search System Operating at Du Pont. J. Chem. Doc. 1968; 8:3–13.

27. Willett P, Winterman V, Bawden D. Implementation of Nearest-Neighbor Searching in an Online Chemical Structure Search System. J. Chem. Inf. Comput. Sci. 1986; 26:36–41.

28. Kristensen TG, Nielsen J, Pedersen CNS. Using Inverted Indices for Accelerating LINGO Calculations. J. Chem. Inf. Model. 2011; 51:597–600. [PubMed: 21332133]

29. Li, C.; Lu, J.; Lu, Y. Efficient Merging and Filtering Algorithms for Approximate String Searches; Proceedings of the 24th International IEEE Conference on Data Engineering (ICDE 2008); 2008. p. 257-266.

30. Vernica R, Li C. Efficient Top-k Algorithms for Fuzzy Search in String Collections. Proceedings of the First International Workshop on Keyword Search on Structured Data. 2009:9–14.

31. Baldi P, Nasr R. When is Chemical Similarity Significant? The Statistical Distribution of Chemical Similarity Scores and Its Extreme Values. J. Chem. Inf. Model. 2010; 50:1205–1222. [PubMed: 20540577]

32. Rogers D, Hahn M. Extended-Connectivity Fingerprints. J. Chem. Inf. Model. 2010; 50:742–754. [PubMed: 20426451]

33. Hassan M, Brown RD, Varma-O'Brien S, Rogers D. Cheminformatics Analysis and Learning in a Data Pipelining Environment. Mol. Diversity. 2006; 10:283–299.

34. Cavnar, WB.; Trenkle, JM. N-Gram-Based Text Categorization; Proceedings of SDAIR 94 3rd Annual Symposium on Document Analysis and Information Retrieval; 1994. p. 161-175.

35. Benz RW, Swamidass SJ, Baldi P. Discovery of Power-Laws in Chemical Space. J. Chem. Inf. Model. 2008; 48:1138–1151. [PubMed: 18522387]

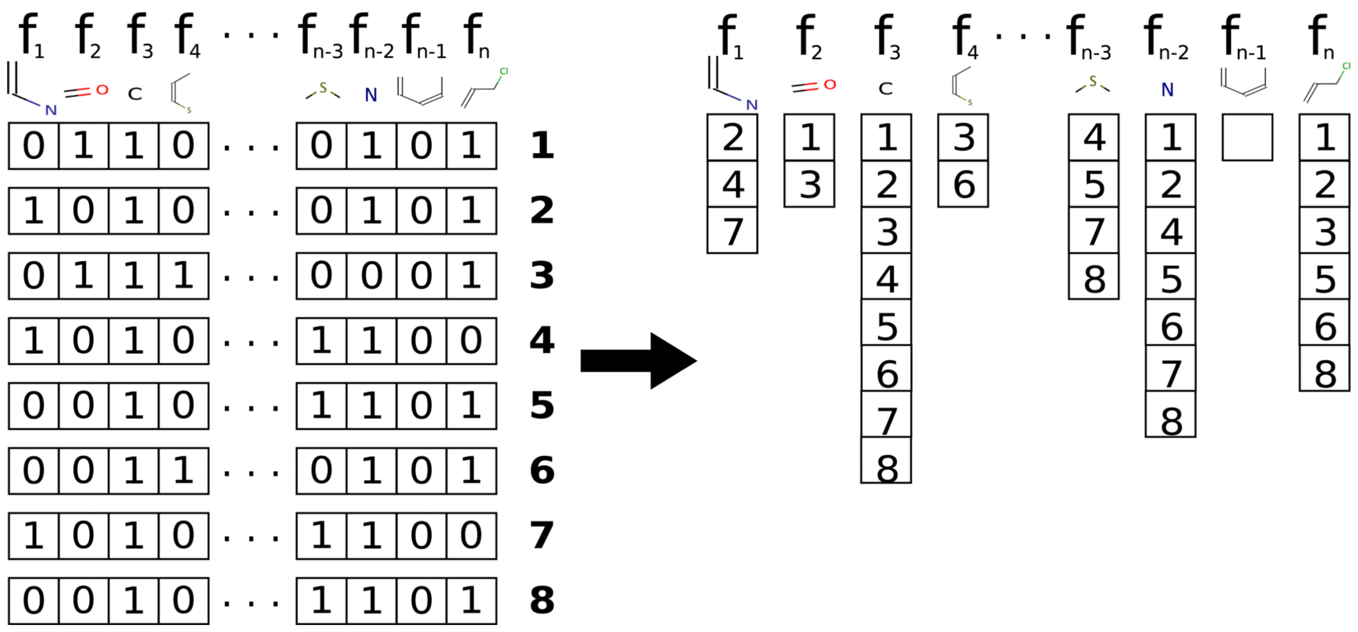

Figure 1.

Inverted Index Data Structure. Each column corresponds to a feature. Each row on the left correspond to a fingerprint. Molecules are numbered from 1 to 8.

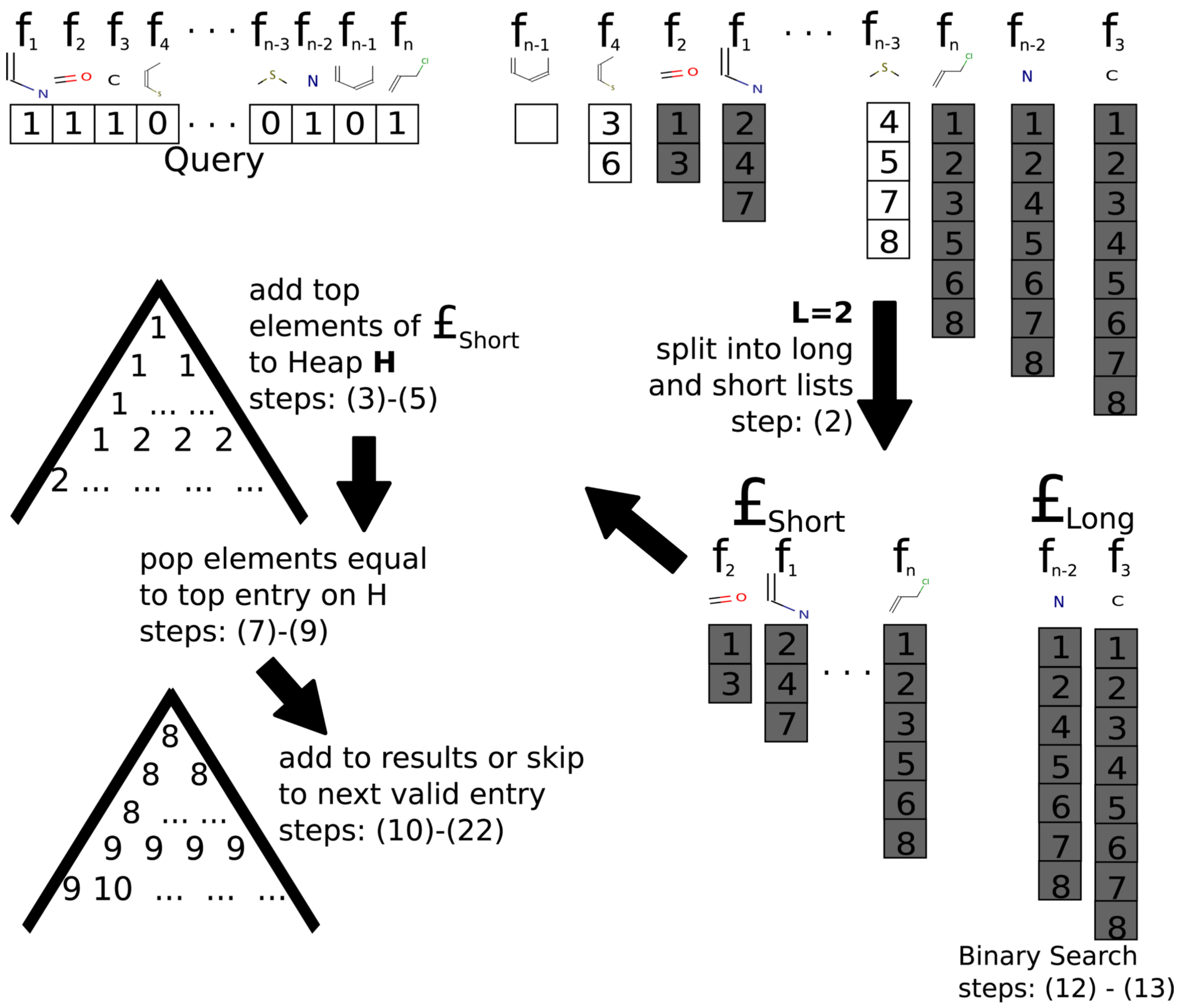
Query
add top elements of £Short to Heap H steps: (3)-(5)
pop elements equal to top entry on H steps: (7)-(9)
add to results or skip to next valid entry steps: (10)-(22)
L=2 split into long and short lists step: (2)
£Short
£Long
Binary Search steps: (12) - (13)

Figure 2.

Diagram of Main Steps of the *DivideSkip* Algorithm with $T = 4$ and $L = 2$. Given a query fingerprint (upper left) and the list of all features ranked by increasing numbers of molecules, only the feature lists corresponding to 1-bits in the query are selected (in gray). The selected feature lists are split into two sets: a set of L longest lists ($£_{Long}$), associated with the features f_{n-2} and f_3, and a set ($£_{Short}$) containing the remaining shorter lists, associated with the features f_2, f_1, and f_n. Any molecule can occur at most $L = 2$ times in $£_{Long}$, and therefore it must occur at least $T - L = 2$ times in $£_{Short}$ to have an intersection of size T or greater with the query. A heap data structure is used to efficiently search for these molecules in $£_{Short}$. In this case, the heap ought to return molecule 1 with 2 counts, molecule 2 with 2 counts, and molecule 3 also with 2 counts. Molecules 1, 2, and 3 and their counts are then used in a binary search of $£_{Long}$ to retrieve the final answer: molecules 1 and 2 are the only molecules with an intersection of size greater or equal to $T = 4$ with the query molecule. In this particular case, both intersections have size exactly equal to $T = 4$. Note that the diagram references step numbers from the pseudocode in Figure 3.

DivideSkip

```
Input: sorted inverted indices lists, $T$, and $L$
Output: a set $R$ of all molecules that occur at least $T$ times

1. Initialize $R$ to an empty result set
2. Split the lists into L longest lists, $£_{Long}$, and the remaining lists, $£_{Short}$
3. Initialize an empty heap $H$
4. FOR each list $\ell$ in $£_{Short}$ {
5.         Insert $\ell[0]$, first entry of $\ell$, into $H$ }
6. WHILE $H$ is not empty {
7.         Let $e$ be the top entry on $H$
8.         Pop all entries in $H$ equal to $e$
9.         Let $n$ be the number of popped entries in (8)
10.        IF $n \geq T - L$ {
11.            FOR each list $\ell'$ in $£_{Long}$ {
12.                    Check if $e$ appears on $\ell'$ (binary search) }
13.            IF $e$ appears at least $T$ times among all lists {
14.                Add $e$ to $R$ }
15.            Insert next entry (if any) from each list into H }
16.        ELSE {
17.            Pop $T - L - 1 - n$ smallest entries from H
18.            Let $e'$ be the current top entry on H
19.            FOR each list $\ell$ {
20.                    Locate its smallest entry $e'' \geq e'$ (binary search)
22.                    Insert $e''$ into $H$ } } }
22. Return $R$
```

Figure 3.
Pseudocode for the Heap Portion of the *DivideSkip* Algorithm.

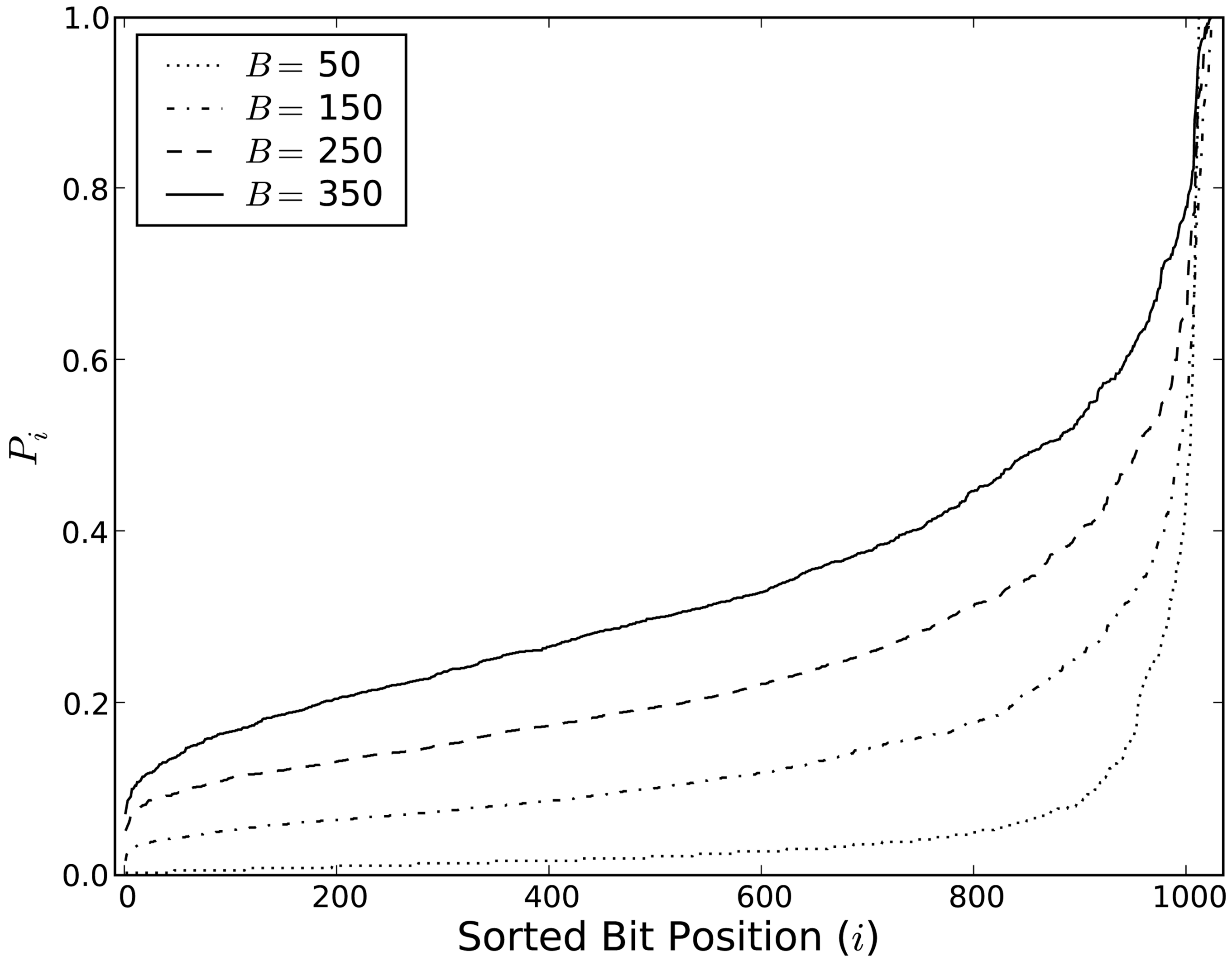

Figure 4.

Fingerprint Component Probabilities. The horizontal axis shows the sorted bit positions i and the vertical axis shows the probability p_i. Each curve corresponds to results of fingerprints sampled with different sizes B, i.e. with different total number of 1-bits. See Empirical Results for a description of the fingerprints used.

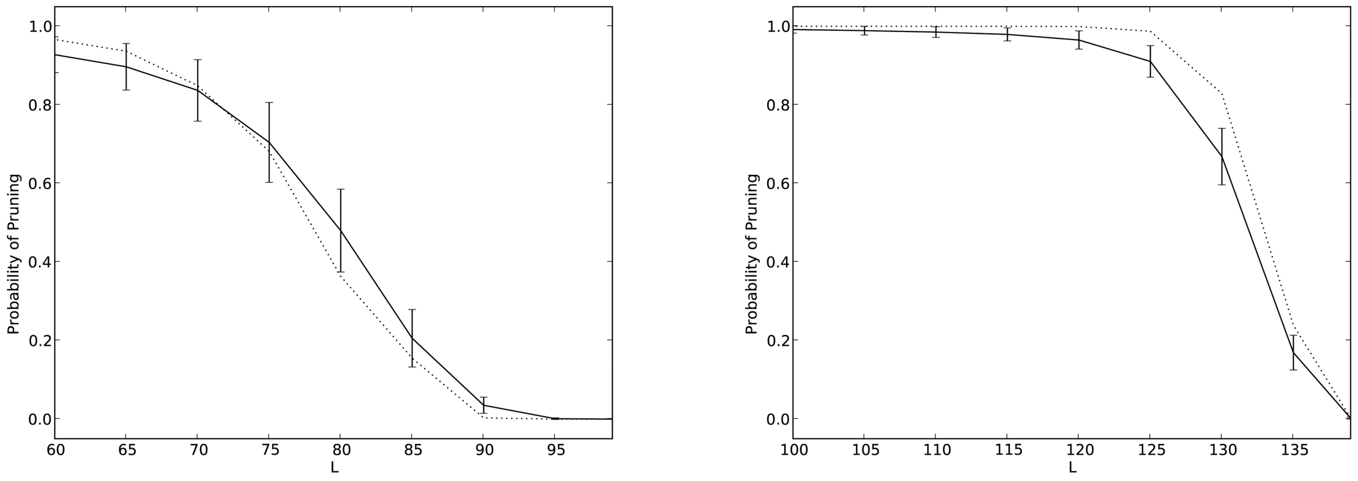

Figure 5.

Comparison of Theoretical and Empirical Probability of Pruning. The curves correspond to a typical query molecule with $A = 200$, in the bin corresponding to $B = 200$, with intersection size $T = 100$ on the left and $T = 140$ on the right. The horizontal axis shows different values of L, and the vertical axis shows the fraction of pruned fingerprints. The solid curve corresponds to the empirical pruning results with a ±1 standard deviation error bars, and the dotted black line represents the theoretical predictions associated with the Poisson distribution (Equation Eq. (6)). The empirical results are obtained from simulations that use 100 queries searched against a 100,000 database sample.

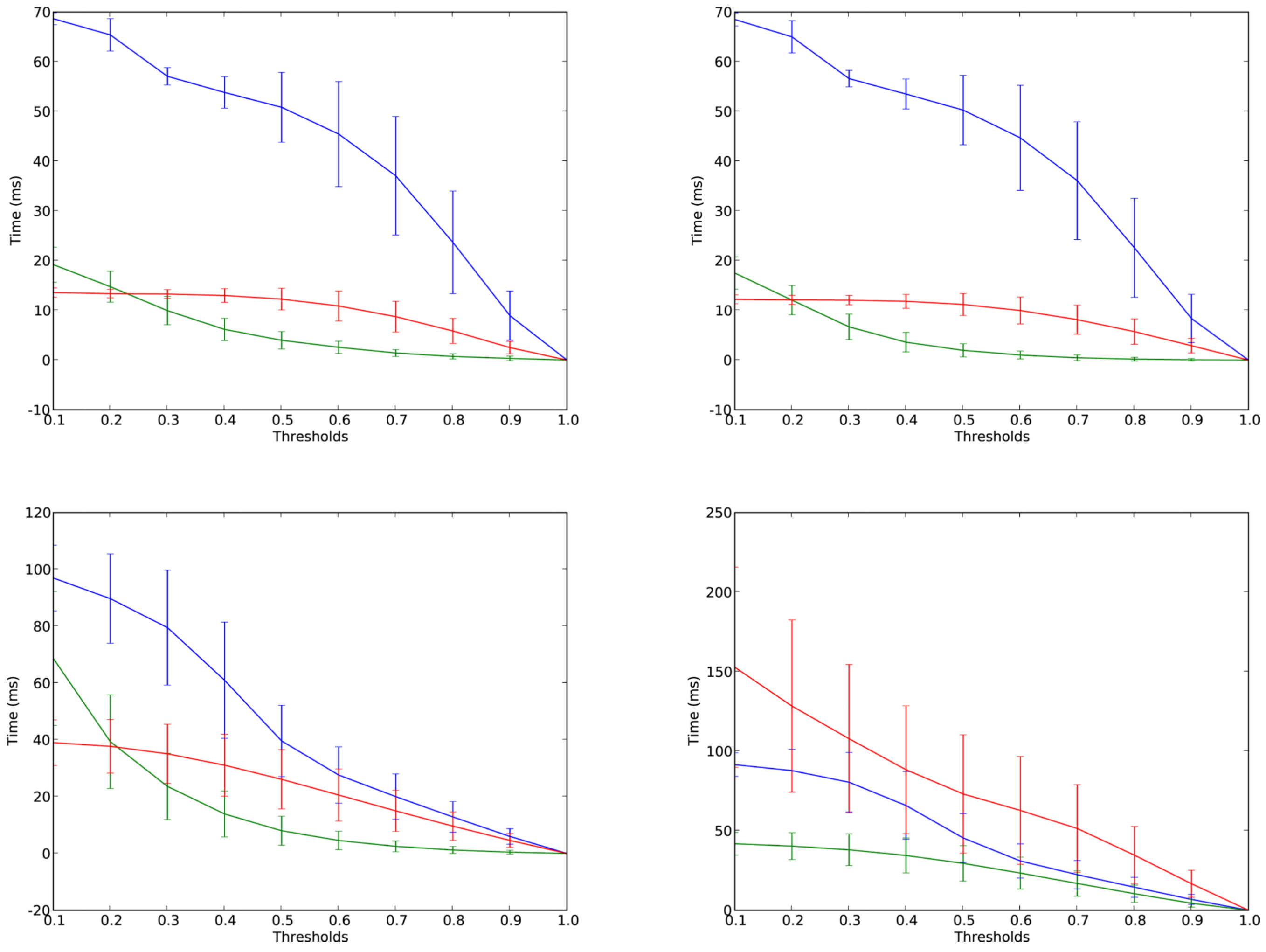

Figure 6.

Timing Results for Threshold Searches. The results correspond to 100 threshold searches against a sample of 100,000 molecules from ChemDB. The means and standard deviations (error bars) of the time in milli-seconds is shown as a function of the Tanimoto threshold. In all sub-figures, the *DivideSkip* method is shown in green, and previous methods, *Hashing* and *MultiBit Tree*, are shown in blue and red respectively. Each sub-figure shows a different combination of features/fingerprints: EC labeling with 1024-modulo-OR fingerprints in the top left, EC labeling with uncompressed fingerprints in the top right, path labeling with 1024-modulo-OR fingerprints in the bottom left, and path labeling with uncompressed fingerprints in the bottom right.

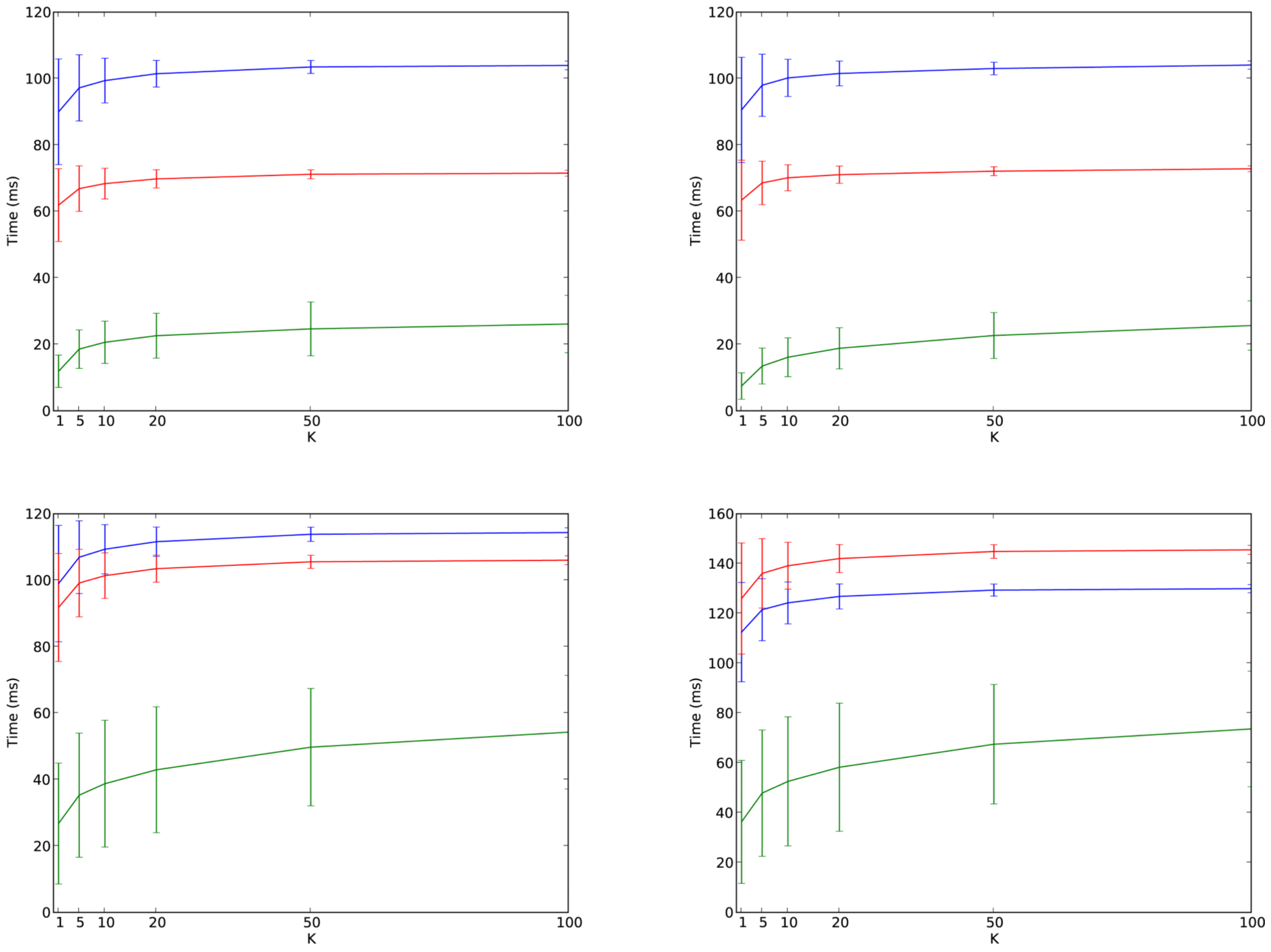

Figure 7.

Timing Results for Top-K Searches. The results correspond to 100 top-K searches against a sample of 100,000 molecules from ChemDB. The means and standard deviations (error bars) of the time in milli-seconds is shown as a function of *K*. In all sub-figures, the *DivideSkip* method is shown in green, and previous methods, *Hashing* and *MultiBit Tree*, are shown in blue and red respectively. Each sub-figure shows a different combination of features/fingerprints: EC labeling with 1024-modulo-OR fingerprints in the top left, EC labeling with uncompressed fingerprints in the top right, path labeling with 1024-modulo-OR fingerprints in the bottom left, and path labeling with uncompressed fingerprints in the bottom right.

Table 1

Mean and Standard Deviation of the Number of Features Present in a Fingerprint. These results correspond to a sample of 100,000 fingerprints from ChemDB. Note that the values differ depending on the type of feature/ fingerprint used. Compressed fingerprints contain less features than the uncompressed fingerprints, and fingerprints based on paths labeling are denser than those based on EC.

Feature type	**paths**		**EC**	
Fingerprint type	**1024-modulo OR**	**uncompressed**	**1024-modulo OR**	**uncompressed**
# of features per fingerprint	218.3 ± 97.7	254.9 ± 136.0	47.0 ± 12.0	48.2 ± 12.5

Table 2

Statistics of Data Sets for Text and Chemical Retrieval. The first row divides the columns into Text and Molecule data. The second row lists the names of the data sets. The third row describes the type of the grams and features. The fourth row shows the number of entries in the data sets, which affects the number of distinct grams/features observed. Finally, the bottom row shows the number of distinct grams observed in text data sets and the number of distinct features observed in the ChemDB data sets.

	Text Data			Molecule Data			
Dataset	**DBLP**	**IMDB**	**Web corpus**	**ChemDB**			
Gram/feature type	**size-3 grams**			**paths**		**EC**	
Size (# of entries)	274,788	1,199,299	2,000,000	100,000	400,000	100,000	400,000
# of grams/features	59,940	34,737	81,620	440,927	756,296	85,390	4,099,828