An Indexing Scheme for Fast and Accurate Chemical Fingerprint Database Searching

Zeyar Aung and See-Kiong Ng

Institute for Infocomm Research
A*STAR (Agency for Science, Technology and Research)
1 Fusionopolis Way, #21-01 Connexis, Singapore 138632
zeyarag@gmail.com, sknq@i2r.a-star.edu.sg

Abstract. Rapid chemical database searching is important for drug discovery. Chemical compounds are represented as long fixed-length bit vectors called fingerprints. The vectors record the presence or absence of particular features or substructures of the corresponding molecules. In a typical drug discovery application, several thousands of query fingerprints are screened for similarity against a database of millions of fingerprints to identify suitable drug candidates. The existing methods of full database scan and range search take considerable amounts of time for such a task. We present a new index-based search method called "ChemDex" (**Chem**ical fingerprint in**Dex**ing) for speeding up the fingerprint database search. We propose a novel chain scoring scheme to calculate the Tanimoto (Jaccard) scores of the fingerprints using an early-termination strategy. We tested our proposed method using 1,000 randomly selected query fingerprints on the NCBI PubChem database containing about 19.5 million fingerprints. Experimental results show that ChemDex is up to 109.9 times faster than the full database scan method, and up to 2.1 times faster than the state-of-the-art range search method for memory-based retrieval. For disk-based retrieval, it is up to 145.7 times and 1.7 times faster than the full scan and the range search respectively. The speedup is achieved without any loss of accuracy as ChemDex generates exactly the same results as the full scan and the range search.

1 Introduction

Chemical database searching plays a crucial role in drug discovery. A group of chemical compounds of interest for a drug are screened (searched) through a database of millions of known compounds to find similar compounds with the desired chemical properties. These compounds can then serve as candidates for the drug to be further analyzed in details. Typically, several thousands of compounds are screened routinely through chemical databases containing millions of fingerprints in developing a single drug [1].

A chemical compound can be represented in three formats: 1-dimensional fingerprint, 2-dimensional structure, and 3-dimensional structure. In the fingerprint representation, a chemical compound is encoded as a fixed-length bit vector (an array of 0's and 1's). Each bit corresponds to the presence (or absence) of a

M. Gertz and B. Ludäscher (Eds.): SSDBM 2010, LNCS 6187, pp. 288–305, 2010.

particular chemical feature such as the existence of certain type of atoms or substructures (e.g. rings). Fingerprint encoding schemes of various lengths are used in the popular chemical databases such as PubChem (881 bits) [2], ChemDB (512 or 1024 bits) [3], and Daylight (2048 bits) [4].

Comparing a pair of 1-dimensional chemical fingerprints is clearly much faster (though less detailed) than comparing a pair of 2- or 3-dimensional chemical structures. As such, screening using fingerprints is routinely used as a first-stage filtering step before going into more comprehensive analysis using 2- or 3-dimensional structures. A number of scoring schemes such as Tanimoto (also known as Jaccard), Tversky, Pearson, Dice, and Kulczynski are available [5,6] for comparing two given fingerprints. Among them, Tanimoto scoring scheme is the most popular and widely used among the chemists [6].

The time incurred for screening thousands of query fingerprints against a database of millions of chemical fingerprints can become quite considerable if each fingerprint is linearly scanned through the chemical database. For instance, while it takes only about 730 nanoseconds to retrieve and compare a single pair of fingerprints on a PC, it takes about 14 seconds to scan a single fingerprint query though the the NCBI PubChem database [2] of 19.5 million fingerprints. For 1,000 queries, it takes about 14,200 seconds (about 4 hours) if the database is to be read repeatedly for every query, and still 4,500 seconds (1 hour and 15 minutes) about even if the database is read only once for all queries. Since a total of tens or even hundreds of thousands of fingerprints may need to be screened against the database in a drug discovery exercise (sometimes interactively), it is important to keep search times to a minimum.

A number of fast search schemes [7,5,8,9,10] have been proposed to reduce the fingerprint screening time (see Section 2 for details). However, all of these methods have their limitations either in terms of accuracy or speed. Among the existing methods, Swamidass and Baldi's range search [5] is the most promising one because it achieves a sub-linear search system with a 100% accuracy.

Our objective is to further reduce the retrieval time of the state-of-the-art Swamidass and Baldi's range search without sacrificing any accuracy. In our approach, we opt to use an indexing scheme, which has been typically used for speeding up the retrievals of various kinds of databases [11].

We propose a novel indexing scheme using an early-termination strategy named "ChemDex" (**Chem**ical fingerprint in**Dex**ing) for rapid searching of chemical fingerprint database. First, we horizontally rearrange (sort) the fingerprints in the database by the total number of 1's they contain. Then, we vertically rearrange (shuffle) the bit order in each fingerprint so that the start region of the fingerprint is generally more populated with 1's than its end region. After the re-arrangements, we sub-divide the fingerprints vertically into k equal-size slices. We then build a two-tier index based on the number of 1's in the whole fingerprint and its fragments in the slices.

When a new fingerprint of a chemical compound (or a family of compounds) is screened against a database, the objective is to rapidly retrieve all the previously characterized compounds contained in the database that are similar to the query

within a user-specified similarity threshold based on a given similarity measure. In this work, we use the industry-standard Tanimoto score [6] to measure the similarity of the fingerprints. Given the user-specified similarity threshold, the search range (lower and upper bounds) is first established. Then, we use a slice-by-slice chain scoring scheme to filter out the unpromising answers. First, all the fingerprint fragments from the first slice within the search range are accessed, and the partial score for each fingerprint is evaluated with the help of the information in the index. For the second slice, only the fragments whose first slice's partial scores are greater than the similarity threshold are accessed. Then, the second slice's partial scores of the accessed fragments are evaluated, and only the qualifying fragments from the third slice are accessed, and so on.

Filtering the fragments slice-by-slice reduces the number of fragments being accessed as we laterally traverse the slices. The total time taken is thus significantly decreased as compared to processing fingerprints in full within the search range. We will formally prove that the partial score for each slice is the upper bound for the final score. In other words, our approach does not leave out any good answer and there is therefore no loss in accuracy with our approach to speed up the search.

As evaluation, we have conducted an experiment of searching 1,000 queries against the NCBI PubChem database [2] of about 19.5 millions fingerprints. For memory-based retrieval, where the whole database is read from disk into memory just once and maintained there throughout all queries, our proposed ChemDex scheme is up to 109.9 times faster than the full database scan, and up to 2.1 times faster than the state-of-the-art Swamidass and Baldi's range search [5] that retrieves the full-length fingerprints within lower and upper bounds. For the traditional disk-based retrieval, where the required portions of the database are read one or more times from disk into memory upon request by the program, ChemDex is up to 145.7 and 1.7 times faster than the full database scan and the range search respectively. As mentioned, ChemDex is 100% accurate as it generates exactly the same result as the full database scan and the range search.

In addition to chemical fingerprint database searching, our proposed bit vector indexing and chain scoring schemes can potentially be used in a range of other applications [12,13,14,15,16] where a large volume of fixed-length bit vectors is involved and Tanimoto score is used to compare them.

2 Related Works

Pairwise sequence comparison methods such as Blast [17] and index-based sequence retrieval methods such as [18] are used in the searching of DNA and protein sequence databases. These methods may be extended to cater for chemical fingerprint bit vectors when they are encoded as character sequences. However, they are primarily designed to deal with variable-length sequences rather than fixed-length ones like chemical fingerprints, and thus are not much suited to handle the latter.

The inverted index search method has been successfully used for searching the fixed-length text document vectors for information retrieval [10]. Since chemical

fingerprints are also fixed-length bit vectors, one can use the inverted index search for the problem of chemical fingerprint retrieval. However, though very effective in processing the text document vectors, our experimental results shows that inverted index search performs poorly for the chemical fingerprint vectors which are relatively much denser with the number of 1's. The performance of the inverted index search for the NCBI PubChem database is reported in Section 4.

Speed is clearly a major concern when designing algorithms for searching chemical databases as we are dealing with large databases. Several researchers have proposed algorithms that traded speed for accuracy. For example, Lim *et al.* [9] suggested a search scheme using an inverted index built on the selected features in the fingerprints. While the method is faster than the full database scan, it is not 100% accurate as only limited set of features are considered. Daylight Inc. [7] has developed a lossy compression method to generate the smaller fingerprints by means of fingerprint folding. Scanning the compressed fingerprint database is faster than scanning the original database. But, this speedup is again achieved with a loss of accuracy.

Baldi *et al.* [8] proposed a lossless compression method to generate the smaller fingerprints and use them for faster database searching. Because the compression is lossless, uncompressed similarity between molecules can be computed exactly from their compressed representations. However, the retrieval time remains linear since all the compressed fingerprints in the database still need to be scanned through.

Swamidass and Baldi [5] have then invented a method that achieves a sublinear retrieval system without any loss of accuracy. Their method reduces the search space by establishing the lower and the upper bounds of the search range. It can be applied either on the original fingerprints or the compressed ones proposed in [8,7]. Yet, even after defining the search range, the number of fingerprints to scan through still remains considerably high. In our proposed ChemDex method, we will further reduce the retrieval time of the Swamidass and Baldi's scheme without sacrificing any accuracy. The performance comparison of the two methods is given in Section 4.

3 Method

3.1 Fingerprint Similarity Measure

We use the most popular industry-standard Tanimoto (also known as Jaccard) similarity score [6] to determine the similarity of two given chemical fingerprints. Tanimoto score S for two fingerprints A and B is calculated as:

$$S(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

where $|A \cap B|$ is the number of 1 bits that are common to both A and B, and $|A \cup B|$ is the number of 1 bits that occur in both or either of A and B. For two 16-bit fingerprints A and B in Figure 1, their Tanimoto score $S(A, B) = 6/11 = 0.5454$. The possible range for a Tanimoto score is from 0 to 1.

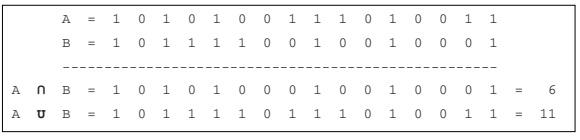

Fig. 1. Calculating Tanimoto score of two fingerprints

In real-life implementations, a bit vector is usually encoded as an array of integers. For example, a 512-bit fingerprint is encoded as an array of 16 32-bit integers. In this case, $|A \cap B|$ can be easily computed using bitwise AND operations on integers and obtaining the number of 1's by a fast bit counting method such as pre-computed bit counting [19]. $|A \cup B|$ can be simply calculated as $|A| + |B| - |A \cap B|$.

Note that in addition to chemical fingerprint comparison, Tanimoto score is also widely used to compare various types of fixed-length binary feature vectors in other applications such as data cleaning [12], all-against-all document similarity [13], stereo image matching [14], video sequence matching [15], embedded software error detection [16], etc. As such, our approach can also be potentially used for speeding up database retrieval in these applications.

3.2 Rearranging Fingerprints Horizontally

In the first step, we rearrange (sort) the order of fingerprints in the database according to the numbers of 1's they contain. This is achieved by an external (disk-based) sorting of fingerprints. The purpose of the horizontal rearrangement is to make the fingerprints ready to be indexed in the future (see Section 3.5). Figure 2 depicts an example of horizontally rearranging fingerprints.

3.3 Rearranging Fingerprints Vertically

Next, we rearrange (shuffle) the bit order of the fingerprints so that the start region of the fingerprint is generally denser with 1's than its end region. In this second step, we first count the frequency of 1's for each bit position (column), and rearrange the order of columns so that the column with the highest frequency of 1's comes first. The purpose of this vertical rearrangement is to improve the filtration power of the lower-order slices — after the database is partitioned into slices as discussed in the next section. (Vertical rearrangement is an indispensable step in our ChemDex method. Without it, the search efficiency of the method is found to be greatly reduced. However, we are not able to show the statistics here because of the space limitation.) An example of vertically rearranging bits is demonstrated in Figure 3.

3.4 Dividing Fingerprints into Slices

After rearranging the fingerprints both horizontally and vertically, we partition them into k equal-size slices vertically. The purpose of partitioning is to enable

Original Fingerprints

																	No. of 1's
Fingerprint #1	1	0	1	0	1	0	0	1	1	1	0	1	0	0	1	1	9
Fingerprint #2	1	0	1	1	1	1	0	0	1	0	0	1	0	0	0	1	8
Fingerprint #3	1	0	0	1	0	1	0	1	1	0	0	0	1	0	1	1	8
Fingerprint #4	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	5
Fingerprint #5	0	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	6
Fingerprint #6	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	7
Fingerprint #7	0	1	1	1	0	1	1	1	1	1	1	0	1	0	1	1	12
Fingerprint #8	1	0	1	1	0	0	1	1	1	1	0	1	0	0	1	1	10

Horizontally Rearranged Fingerprints

																	No. of 1's
Fingerprint #4	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	5
Fingerprint #5	0	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	6
Fingerprint #6	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	7
Fingerprint #2	1	0	1	1	1	1	0	0	1	0	0	1	0	0	0	1	8
Fingerprint #3	1	0	0	1	0	1	0	1	1	0	0	0	1	0	1	1	8
Fingerprint #1	1	0	1	0	1	0	0	1	1	1	0	1	0	0	1	1	9
Fingerprint #8	1	0	1	1	0	0	1	1	1	1	0	1	0	0	1	1	10
Fingerprint #7	0	1	1	1	0	1	1	1	1	1	1	0	1	0	1	1	12

Fig. 2. Rearranging fingerprints horizontally

Original Fingerprints

Original Bit Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Fingerprint #4	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0
Fingerprint #5	0	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0
Fingerprint #6	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0
Fingerprint #2	1	0	1	1	1	1	0	0	1	0	0	1	0	0	0	1
Fingerprint #3	1	0	0	1	0	1	0	1	1	0	0	0	1	0	1	1
Fingerprint #1	1	0	1	0	1	0	0	1	1	1	0	1	0	0	1	1
Fingerprint #8	1	0	1	1	0	0	1	1	1	1	0	1	0	0	1	1
Fingerprint #7	0	1	1	1	0	1	1	1	1	1	1	0	1	0	1	1
Frequency	6	1	5	5	4	3	3	7	8	4	2	5	3	0	4	5
Vertically Rearranged Fingerprints																
Fingerprint #4	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0
Fingerprint #5	1	1	0	1	0	0	0	0	1	0	0	1	0	1	0	0
Fingerprint #6	1	1	1	0	1	1	0	1	0	0	0	0	1	0	0	0
Fingerprint #2	1	0	1	1	1	1	1	1	0	0	1	0	0	0	0	0
Fingerprint #3	1	1	1	0	1	0	1	0	0	1	1	0	1	0	0	0
Fingerprint #1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0
Fingerprint #8	1	1	1	1	1	1	1	0	1	1	0	1	0	0	0	0
Fingerprint #7	1	1	0	1	1	0	1	0	1	1	1	1	1	1	1	0
Rearranged Bit Order	9	8	1	3	4	12	16	5	10	15	6	7	13	11	2	14

Fig. 3. Rearranging fingerprints vertically

retrieval and processing of smaller fingerprint fragments rather than the full-length fingerprints.

3.5 Building Index

For fast retrieval, a two-tier index is constructed. The entries in the first tier correspond to the number of 1's in the fingerprints. For an m-bit fingerprint, the dimension of the first tier is $m + 1$ (0 to m). Each entry in the first tier points to the records in the second tier having that number of 1's. Since the number of fingerprints in large databases (usually millions or more) is much larger than the number of bits in a fingerprint (hundreds to thousands), a majority of first tier entries each points to multiple records in the second tier.

Each record in the second tier contains an abstract of the information of an individual fingerprint: its ID, the number of 1's in the full-length fingerprint, and the number of 1's in the fingerprint fragments in each of its slices. The number of records in the second tier is the same as the number of fingerprints in the database. Each second tier record points to the fingerprint fragments in the slice files which are stored on disk or in memory.

On an average modern PC, both the first and the second tiers of the index for a large database (like the NCBI PubChem database of 19.5 million fingerprints) can be stored in memory. Although the second tier stores the information for every fingerprint in the database, its record size is much smaller than that of an actual fingerprint. For example, while the size of an 881-bit PubChem fingerprint is 112 bytes, the size of a second tier entry for a 2-slice partitioning is only 10 bytes, and that for a 4-slice partitioning is only 14 bytes. Thus, for the PubChem

database, the size of the second tier for a 2-slice (4-slice respectively) partitioning will be 186 MB (260 MB respectively), which can be conveniently stored in a modern PC with a memory capacity of 1 GB or above.

A small example of the two-tier index with 2-slice partitioning is illustrated in Figure 4.

3.6 Query Evaluation

When a query fingerprint Q is submitted together with a similarity threshold t, its bit order is rearranged in the same way as the fingerprints in the database (see Section 3.3), and it is divided into k fragments as those in the database (see Section 3.4). Then, the number of 1's in it, $|Q|$, is counted, and the lower and the upper bounds of the number of 1's for the database's fingerprints to match with the query are calculated. We can compute these bounds based on the equations given in Swamidass and Baldi [5].

$$Lower(Q,t) = ceiling(|Q| \times t) \tag{2}$$

$$Upper(Q,t) = floor(|Q| \,/\, t) \tag{3}$$

For example, if the number of 1's in the query is 7 and the similarity threshold is 0.75, its lower bound is $ceiling(7{\times}0.75) = 6$, and its upper bound is $floor(7 \,/\, 0.75) = 9$. That means we only have to look at the database's fingerprints with their numbers of 1's between 6 and 9, and simply ignore the others.

Now, let us further reduce the amount of data to be accessed within the upper and the lower bounds using an early-termination approach. We propose a chain scoring scheme that incrementally calculates a fingerprint's partial score for each slice up to the final score in the last slice. If the partial score at a certain slice is less then the given similarity threshold, this fingerprint is discarded without having to access the remaining slices any further.

First, we access the fingerprint fragment from Slice #1, and calculate its partial score with the help of the information stored in the index second tier. For a query fingerprint Q and a target fingerprint F, their partial score $S_1(Q,F)$ for Slice #1 is calculated as:

$$S_1(Q,F) = \frac{|Q_1 \cap F_1| + \sum_{j=2}^{k} \min(|Q_j|, |F_j|)}{|Q_1 \cup F_1| + \sum_{j=2}^{k} \max(|Q_j|, |F_j|)} \tag{4}$$

where Q_1 and F_1 are Slice #1's fingerprint fragments of the query and the target respectively, and $|Q_j|$ and $|F_j|$ are the number of 1's in their remaining slices $2..k$ respectively. $|Q_j|$ and $|F_j|$ can be readily obtained from the index's second tier.

Since Slice #1's partial score assumes the maximum possible matches in each of the remaining slices $2..k$, it is the upper bound of the actual final score. (We will prove this in Lemma 1.) Consequently, if a partial score of a fingerprint is less than the similarity threshold, its final score can never be greater than or equal to that threshold. This means that accuracy is maintained as we will not be losing any of the good answers. (We will prove this in Theorem 1.)

Thus, when we access the fingerprint fragments from Slice #2, only the qualified ones with their partial scores greater than or equal to the similarity threshold need to be examined. The unqualified ones can be discarded immediately without having to look at its fingerprint fragments in Slice #2 and the later slices. For the qualified ones, their Slice #2's partial scores are calculated, and the process is carried on only for those qualified ones to Slice #3, and so on. In this way, we repetitively reduce the number of fingerprint fragments to be retrieved from each slice as we laterally traverse the slices. The partial scores for the slices $2..k-1$ are calculated in the same manner as that for Slice #1.

$$S_i(Q,F) = \frac{|Q_{1..i} \cap F_{1..i}| + \sum_{j=i+1}^{k} \min(|Q_j|, |F_j|)}{|Q_{1..i} \cup F_{1..i}| + \sum_{j=i+1}^{k} \max(|Q_j|, |F_j|)} \tag{5}$$

where $2 \leq i \leq k-1$.

The partial score $S_k(Q,F)$ for the last slice k is the essentially the final score $S(Q,F)$ of the fingerprint F.

$$S(Q,F) = S_k(Q,F) = \frac{|Q_{1..k} \cap F_{1..k}|}{|Q_{1..k} \cup F_{1..k}|} \tag{6}$$

Let us now prove the upper bound property of the partial scores.

Lemma 1. *For a query fingerprint Q and a target fingerprint F, their partial score $S_i(Q,F)$ for any slice i ($1 \leq i \leq k-1$, where k is the total number of slices a fingerprint is partitioned) is the upper bound for the final score $S(Q,F)$.*

$$S_i(Q,F) \geq S(Q,F)$$

Proof. Without loss of generality, let us take $i = 1$ and look at the partial score $S_1(Q,F)$ for Slice #1.

Since the size of the intersection of two sets is at most that of the smaller set, for some non-negative integer x_j $(2 \leq j \leq k)$, we have:

$$\min(|Q_j|, |F_j|) \geq |Q_j \cap F_j| \Leftrightarrow \min(|Q_j|, |F_j|) = |Q_j \cap F_j| + x_j$$

Again, since the size of the union of two sets is at least that of the larger set, for some non-negative integer y_j $(2 \leq j \leq k)$, we have:

$$\max(|Q_j|, |F_j|) \leq |Q_j \cup F_j| \Leftrightarrow \max(|Q_j|, |F_j|) = |Q_j \cup F_j| - y_j$$

We can then rewrite Equation 4 as:

$$S_1(Q,F) = \frac{|Q_1 \cap F_1| + \sum_{j=2}^{k}(|Q_j \cap F_j| + x_j)}{|Q_1 \cup F_1| + \sum_{j=2}^{k}(|Q_j \cup F_j| - y_j)}$$

$$= \frac{|Q_1 \cap F_1| + \sum_{j=2}^{k}|Q_j \cap F_j| + \sum_{j=2}^{k} x_j}{|Q_1 \cup F_1| + \sum_{j=2}^{k}|Q_j \cup F_j| - \sum_{j=2}^{k} y_j}$$

But, since x_j and y_j $(2 \leq j \leq k)$ are non-negative integers, we have:

$$\sum_{j=2}^{k} x_j \geq 0 \text{ and } \sum_{j=2}^{k} y_j \geq 0$$

Again, since subtracting a non-negative integer from the quotient and adding another non-negative integer to the divisor of a non-negative fractional number makes the resultant number smaller than or equal to the original number, we finally have:

$$S_1(Q,F) \geq \frac{|Q_1 \cap F_1| + \sum_{j=2}^{k} |Q_j \cap F_j|}{|Q_1 \cup F_1| + \sum_{j=2}^{k} |Q_j \cup F_j|} = \frac{|Q_{1..k} \cap F_{1..k}|}{|Q_{1..k} \cup F_{1..k}|} = S(Q,F)$$

Similarly, $S_i(Q,F) \geq S(Q,F)$ for the other slices $i = 2$ to $k-1$.

Now, we will prove the accuracy property of the filtration by the partial scores.

Theorem 1. *Any filtration using the partial scores $S_i(Q,F)$ $(1 \leq i \leq k-1)$ does not discard any good answer.*

Proof. Given the similarity threshold t, at any slice i, we filter out a fingerprint F which satisfies the condition:

$$S_i(Q,F) < t$$

From Lemma 1 we have:

$$S(Q,F) \leq S_i(Q,F)$$

Thus, we finally have:

$$S(Q,F) \leq S_i(Q,F) \ \wedge \ S_i(Q,F) < t \ \Rightarrow S(Q,F) < t$$

This means if the partial score of a fingerprint is less than the similarity threshold, its final score will also be less than this threshold, and we can safely reject it. Thus, filtration using the partial scores will not discard any good answer whose final score is greater than or equal to the similarity threshold. In other words, a 100% accuracy is guaranteed.

An example of evaluating a query is demonstrated in Figure 4. The records that are accessed are highlighted in gray. The fragments for Fingerprint IDs #5, #6, #2, #3, and #1 are accessed from Slice #1, and those for Fingerprint IDs #2 and #1 are accessed from Slice #2. Only Fingerprint ID #2 is returned as the answer as its score is greater than 0.75.

Suppose the size of a full-length fingerprint is l bytes. For our example, if the full database scan is used, $8l$ bytes will be processed (i.e. accessed from memory/disk and compared to the query). If the range search method [5] is used, $5l$ bytes will be processed. Using our ChemDex method, only 7 half-length fingerprint fragments of size $7 \times 0.5l = 3.5l$ bytes are processed. Thus, we have improved the speed by 8 / 3.5 = 2.29 times compared to the full database scan, and 5 / 3.5 = 1.43 times compared to the range search in this example.

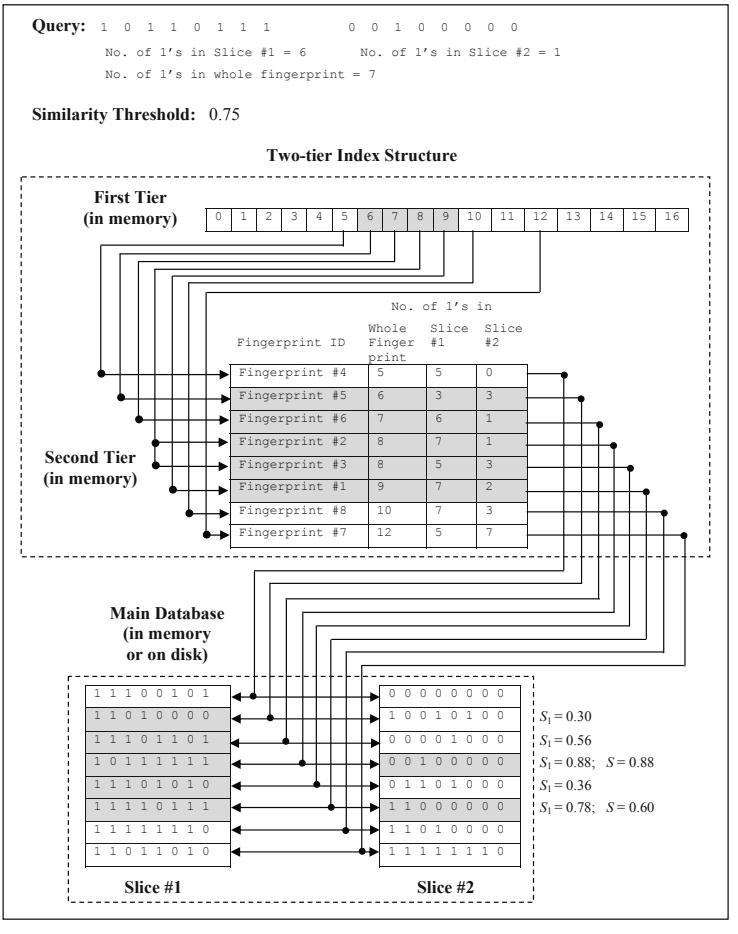

Fig. 4. ChemDex's two-tier index and evaluation of a query using it. The records that are accessed are highlighted in gray.

4 Results and Discussions

4.1 Experimental Setup

We use the PubChem [2] database from NCBI (National Center for Biotechnology Information, USA) downloaded in November 2008. It contains 19,501,867 (about 19.5 millions) fingerprints. The length of a fingerprint is 881 bits. We encode the fingerprint into an array of 28 32-bit integers. ($28 \times 32 = 896$. Since we only have 881 bits, the last 15 bits of the last integer are used as padding.)

The statistics of the number of 1's among the PubChem fingerprints is as follows: the minimum is 0 (i.e. some fingerprints have no 1's at all), the maximum is 290, the median is 146, the mode is 156, the mean is 139.71, and the

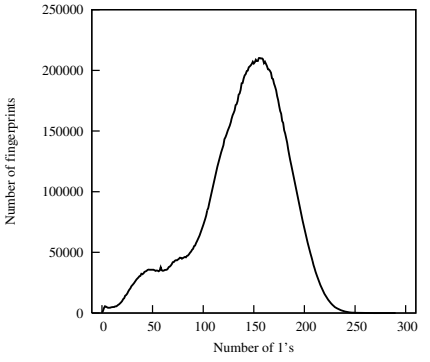

Fig. 5. Distribution of the number of 1's in the PubChem database

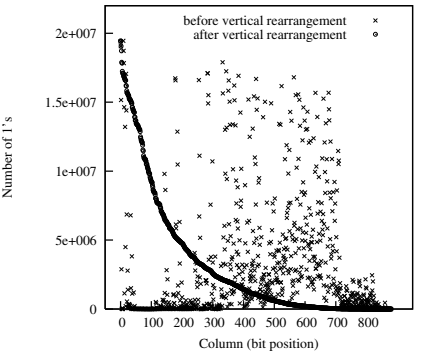

Fig. 6. Distribution of the number of 1's in columns before and after vertical rearrangement

standard deviation is 42.58. The distribution of the number of 1's in the PubChem fingerprints is given in Figure 5.

The statistics of the number of 1's in different bit positions (columns) is as follows: the minimum is 1 (i.e. only 1 out of 19,501,867 bits is set to 1 in the least populated column), the maximum is 19,450,390 (i.e. 99.74% of the bits are set to 1 in the most populated column), the mean is 3,092,714.87, and the standard deviation is 4,431,637.98. The distribution of the number of 1's in the columns before and after the vertical rearrangement (Section 3.3) is given in Figure 6.

From the database, we randomly select 1,000 fingerprints to serve as queries. As the mean of number of 1's in the query fingerprints is 138.69 and the standard deviation is 42.90, we can say that the query data set has a similar distribution of the number of 1's as the entire database.

We search these 1,000 queries against the PubChem database of 19,501,867 fingerprints using 7 different Tanimoto similarity thresholds: 1.00, 0.95, 0.90, 0.85, 0.80, 0.75, and 0.70. We do not include the similarity thresholds lower than 0.70 in our experiments. The lower thresholds are not useful for real-life drug discovery applications as they tend to generate large volumes of results which are impractical to be further analyzed in a meaningful way. For reference, the PubChem's search website [20] uses an even higher value of 0.80 as the lowest possible threshold value.

We run our experiments on a standard PC with Intel Pentium D 2.8 GHz CPU, 4 GB RAM, and 300 GB SATA2 disk drive, running the 32-bit version of Windows Vista. All the programs are written in C++ and compiled with Microsoft Visual C++ 2008 Express Edition using the maximize speed (`/O2`) optimization. We execute the programs with high system priority (using `start /high` command), and make sure that no background service such as anti-virus or Windows file indexing is running during the executions of the programs. We also make certain that every data file involved in the experiment is in a single disk fragment by using JKDefrag [21].

We will consider the problem of fingerprint database retrieval in two scenarios: memory-based and disk-based. For each scenario, we search the 1,000 query fingerprints against the PubChem database using five methods: (1) full database scan (baseline), (2) inverted index search [10], (3) Swamidass and Baldi's range search [5], (4) ChemDex with 2-slice partitioning, and (5) ChemDex with 4-slice partitioning.

4.2 Memory-Based Retrieval

In the memory-based scenario, we have enough memory to hold all the data file(s) that are required to evaluate the queries. When processing multiple queries as a batch, all the required data files are loaded from disk into memory only once, and are maintained there throughout the processing of all queries.

For the full database scan method, the whole PubChem database (2,083 MB) is loaded into memory. This takes an average of 9 seconds.

For the Swamidass and Baldi's range search method [5], the whole fingerprint database plus an index file indicating the the number of 1's in the fingerprints (a total of 2,194 MB) are loaded. This takes an average of 10 seconds.

For ChemDex with 2-slice partitioning, 2 half-length fingerprint fragment files plus the first tier and the second tier index files (a total of 2,269 MB) are loaded. This takes an average of 10 seconds.

For ChemDex with 4-slice partitioning, 4 quarter-length fingerprint fragment files plus the first tier and the second tier index files (a total of 2,343 MB) are loaded. This takes an average of 10 seconds.

For the inverted index search method, the size of the inverted index for the entire PubChem database is 10,394 MB, which is too big to be held in our PC with 4,096 MB (4 GB) of memory. Thus, we split the original database into 4 smaller databases of approximately equal sizes, build 4 smaller inverted indexes, load each index at a time and use it for the processing of all the 1,000 queries, and finally consolidate the results form 4 inverted indexes. The total time taken to load the four inverted indexes into memory is 47 seconds. (In fact, we have also tried variable-byte encoding [22] to compress the inverted index into 2,609 MB which can be fit into the memory. However, because of the overhead of decompression incurred when accessing the compressed index, the searching time is even longer by a factor of 1.5. We do not show the results of the compressed index for the sake of clarity.)

The running times of the five methods for this memory-based retrieval exercise are presented in Figure 7. Note that each time figure does not include the one-time cost of loading the relevant data file(s) from disk into memory. For all the five methods, care was taken such that none of the data is read more than once from disk, even in the case of the inverted index search method where we had to split the database into 4 smaller databases.

For each method, the average volume of data processed (accessed from memory and compared with their respective counterparts in the query to calculate the Tanimoto score) for a query is shown in Figure 8.

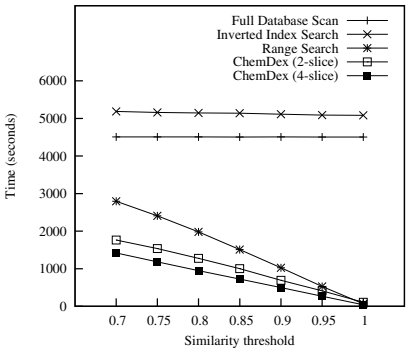

Fig. 7. Running times for searching 1,000 queries though the PubChem database for memory-based retrieval

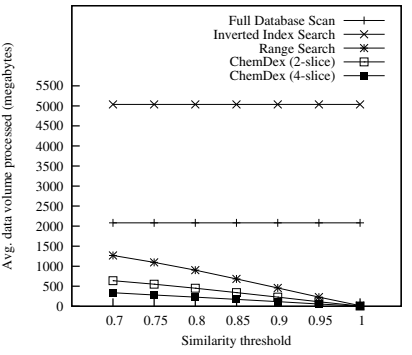

Fig. 8. Average volumes of data to be processed for a query

Except for the inverted index search, the distribution of the times taken for memory-based retrieval (Figure 7) is more or less proportionate to the distribution of the data volumes that is processed (Figure 8). This is because the time taken to access any fingerprint (or a fragment of it) from memory is virtually constant. As such, the 4-slice version of ChemDex gives consistently better results than its 2-slice version does, since a smaller data volume needs to be accessed and processed in the former. Therefore, the best strategy for ChemDex in memory-based retrieval is to always use the 4-slice partitioning.

When compared with the full database scan, ChemDex (4-slice version) is clearly much faster in all cases. In the best case, it is 109.88 times faster than the full scan (for the similarity threshold $t = 1.00$). In the worst case, it is still 3.18 times faster (for $t = 0.70$).

In comparison with the range search, ChemDex (4-slice version) is 2.09 times faster than it in the best case (for $t = 0.80$), and still 1.66 time faster than it (for $t = 1.00$) in the worst case.

In relation to the inverted index search, ChemDex (4-slice version) is 123.93 times faster than it in the best case (for $t = 1.00$), and still 3.65 time faster than it (for $t = 0.70$) in the worst case.

It is interesting to learn here that the performance of the inverted index search is even worse than that of the full database scan. While inverted index search was reported to perform well in other areas like text database retrieval [10], our experiments have revealed it to be unsuitable for the task of chemical fingerprint retrieval. This is due to the relatively dense nature of the chemical fingerprints. The average density of 1's in PubChem's fingerprints with respect to their length is 139.71/ 881 = 0.1586 = 15.86%, whereas the average densities of the document vectors in text databases are normally much lower (for example 0.86% for MEDLINE medical abstract database and 1.82% for Time Magazine's news abstract database [23]). Thus, a large volume of information needs to be accessed from all the inverted lists corresponding to the 1's in the query. For one query,

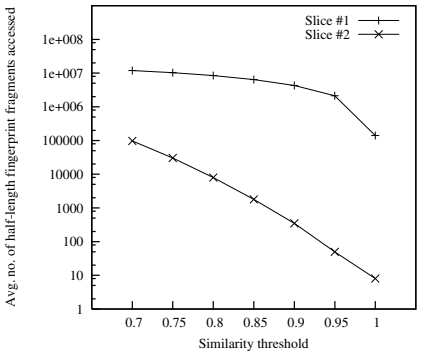

Fig. 9. Average number of half-length fingerprint fragments accessed from Slices #1 and #2 in 2-slice version of ChemDex

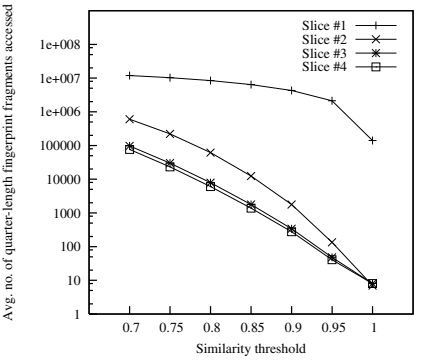

Fig. 10. Average number of quarter-length fingerprint fragments accessed from Slices #1, #2, #3 and #4 in 4-slice version of ChemDex

the average amount of inverted list information required to be accessed from memory is 5,035 MB, which is 2.4 times of the 2,083 MB fingerprint information needed to be accessed for the full database scan (The overall difference in their running times is less than 2.4 times because the full database scan involves a relatively expensive operation of bit counting, whereas the inverted index search does not.)

Let us investigate the filtration power of ChemDex. Figures 9 and 10 show the average numbers of half-length and quarter-length fingerprint fragments accessed from the different slices for the 2-slice and the 4-slice versions of ChemDex respectively (do note that the Y-axis in each figure is in log-scale). It can be observed in Figure 9 that the number of fingerprint fragments accessed from Slice #2 is only a small fraction of that of Slice #1. This means that a large number disqualified fingerprints have been filtered out from the processing of Slice #1. Similarly, the numbers of fragments accessed from Slice #2, #3, #4 are only small fractions of that of Slice #1 in Figure 10. As a result, the total volumes of data required to be accessed by ChemDex (both for 2-slice and 4-slice versions) is much lower that of Swamidass and Baldi's range search which needs to access all the full-length fingerprints within the specified search range.

4.3 Disk-Based Retrieval

Since the size of a chemical database can potentially grow up to 10^{60} compounds [24], it is possible in the future that the growth of the database size outpaces the amount of memory that is generally affordable. When the size of the database becomes too large to fit into memory, we have to employ a traditional disk-based retrieval approach. In disk-based retrieval, all the data file(s) are primarily stored on disk, and only the required pieces of data are read from disk into memory as requested by the program. For example, in the full database

scan, for a particular query, a database fingerprint at a time is read from disk into memory to compare with the query. For another query, the same fingerprint will have to be read again from disk.

In addition to the limited memory situation, the disk-based scenario is also applicable to a situation in which queries are necessary to be processed one-by-one due to their different arrival times (as opposed to the batch processing of the queries as discussed in the above Section 4.2). This is particularly important if the fingerprint database search program is to be run on a general office PC or laptop rather than on a dedicated server. It is obviously not desirable to hold a large database in memory when not in use, since this can adversely affect the performances of the other applications running on the same machine.

Their running times the four different methods are presented in Figure 11. (The result of the inverted index search again turns out to be even worse than that of the full database scan, and hence is excluded for the sake of clarity.)

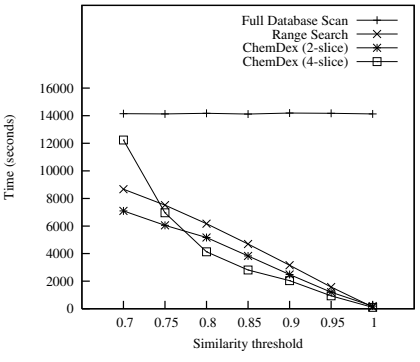

Fig. 11. Running times for searching 1,000 queries though the PubChem database for disk-based retrieval

ChemDex achieves the best results using the 4-slice partitioning for the similarity thresholds of 0.80–1.00, and with the 2-slice partitioning for the thresholds of 0.70–0.75. Thus, the optimization of ChemDex for the disk-based retrieval can be achieved by keeping the indexes and database files for both the 2-slice and the 4-slice versions, and using the 2-slice one if the user-specified similarity threshold is less than 0.8, and the 4-slice one otherwise.

ChemDex, using the above optimization, is clearly much faster the full database scan in all cases. In the best case, it is 145.71 times faster than the full scan (for the similarity threshold $t = 1.00$). In the worst case, it is still 1.99 times faster (for $t = 0.70$).

In comparison with the range search, ChemDex (using the optimization) is 1.69 times faster than it in the best case (for $t = 0.95$), and still 1.22 times faster than it (for $t = 0.70$) in the worst case.

It can be noticed from Figure 8 that the volume of data to be processed for ChemDex (for both the 2-slice and 4-slice versions) is much lower than that of

the range search. However, unlike memory-based retrieval, the time incurred in processing the data here is not as proportionately reduced. (Compare Figure 8 with Figure 11.) This is because the range search can read in all data blocks containing the full-length fingerprints from disk consecutively, whereas ChemDex can only read in Slice #1 in the consecutive manner. For the remaining slices, the fingerprint fragments to be read in can be sparsely located, and ChemDex cannot benefit from the effect of disk buffering in accessing them. As a result, the time taken to access the fingerprint fragments in the later slices can be relatively high compared to that for accessing the sequential fragments in Slice #1.

The uptick in the response time observed on the curve for ChemDex (4-slice) in Figure 11 is because of the dramatic surge in the number of sparse quarter-length fragments to be read from Slices #2, #3, and #4 for the lower thresholds of 0.70–0.75. (See Figure 10 whose Y-axis is in log-scale.)

4.4 Preprocessing Costs

The proposed ChemDex method requires some preprocessing steps to rearrange the fingerprints and to build the index. Table 1 shows the preprocessing times for the PubChem database using 2-slice and 4-slice partitionings.

Table 1. ChemDex's preprocessing times for the PubChem database

Step	Procedure	Time (seconds)	
		2-slice	4-slice
1	Rearranging fingerprints horizontally (Section 3.2)	835	835
2	Rearranging fingerprints vertically (Section 3.3)	1,937	1,937
3	Dividing fingerprints into slices (Section 3.4)	205	316
4	Building Index (Section 3.5)		
	(a) First tier	84	84
	(b) Second tier	78	126
	Total	**3,139**	**3,298**

Although the full database scan does not require any preprocessing and the range search method requires only Steps #1 and #4(a) of preprocessing, it should be noted that ChemDex's preprocessing costs are only one-time costs. So, when thousands of queries are evaluated routinely and/or interactively, the time savings by ChemDex overrides the time spent in the preprocessing.

5 Conclusion

In this paper, we have presented a chemical fingerprint database search system named "ChemDex". ChemDex uses an index-based early-termination approach

in which the large chemical fingerprint database is both horizontally and vertically rearranged, partitioned into slices, and then indexed. A chain scoring scheme is then used for query evaluation: the fingerprint fragments from the initial slices are retrieved and their partial scores (upper bounds of their respective final scores) are calculated with the information stored in the index; the process continues into the subsequent slices only if the computed partial score is greater than or equal to the given similarity threshold. This chain scoring scheme enables the search space to be pruned effectively without any loss of accuracy.

We have tested our proposed method using the NCBI PubChem database with about 19.5 million fingerprints and 1,000 randomly selected queries. Our experimental results show that in the memory-based scenario, ChemDex is up to 109.9 times faster than the full database scan and up to 2.1 times faster than the state-of-the-art range search. In the traditional disk-based scenario, ChemDex is up to 145.7 times and 1.7 times than the full database scan and the range search respectively. ChemDex is able to achieve these speedups while maintaining the same level (100%) of accuracy as the slower methods.

Methods such as ChemDex is important for drug discovery as the chemical compound database search space can potentially grow considerably beyond its current typical value of a few million molecules towards the estimated 10^{60} size of the virtual space of small organic molecules [24]. In addition, the proposed bit vector indexing and chain scoring schemes in ChemDex can also be applicable in other domains whereby very large databases of fixed-length bit vectors are involved. As future work, we will explore some such other applications as document processing, image and video processing, error detection, etc., where the proposed techniques can also be exploited.

References

1. Alvarez, J., Shoichet, B. (eds.): Virtual Screening in Drug Discovery. CRC Press, Boca Raton (2005)
2. The PubChem Project, `http://pubchem.ncbi.nlm.nih.gov/`
3. ChemDB.com Chemistry Databases, `http://cdb.ics.uci.edu/`
4. Daylight Chemical Information Systems Inc., `http://www.daylight.com/`
5. Swamidass, S.J., Baldi, P.: Bounds and algorithms for fast exact searches of chemical fingerprints in linear and sub-linear time. J. Chem. Info. Model. 47, 302–317 (2007)
6. Willett, P.: Similarity-based virtual screening using 2D fingerprints. Drug Discov. Today 11, 1046–1053 (2006)
7. Daylight User Manual, `http://www.daylight.com/dayhtml/doc/theory/index.html`
8. Baldi, P., Benz, R.W., Hirschberg, D.S., Swamidass, S.J.: Lossless compression of chemical fingerprints using integer entropy codes improves storage and retrieval. J. Chem. Info. Model. 47, 2098–2109 (2007)
9. Lim, C., Chua, S.L., Bong, X.F., Lim, C.G., Lau, S.Y.H., Wang, E.P., Chung, K.Y., Wong, V.P.Y., Aung, Z.: Rapid chemical database search: a filter-and-refine approach. In: Poster Proceedings of the 18th International Conference on Genome Informatics, pp. 90–91 (2007)

10. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
11. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann Publishers, San Francisco (2006)
12. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: Proceedings of the 17th International World Wide Web Conference, pp. 131–140 (2008)
13. Bayardo, R.J., Ma, Y., Srikant, R.: Scaling up all pairs similarity search. In: Proceedings of the 16th International World Wide Web Conference, pp. 131–140 (2007)
14. Cyganek, B.: Comparison of nonparametric transformations and bit vector matching for stereo correlation. In: Klette, R., Žunić, J. (eds.) IWCIA 2004. LNCS, vol. 3322, pp. 534–547. Springer, Heidelberg (2004)
15. Ren, W., Singh, S.: Video sequence matching with spatio-temporal constraints. In: Proceedings of the 17th International Conference on Pattern Recognition, pp. 834–837 (2004)
16. Zoeteweij, P., Abreu, R., Golsteijn, R., van Gemund, A.J.C.: Diagnosis of embedded software using program spectra. In: Proceedings of the 14th Annual IEEE International Conference & Workshops on Engineering of Computer-Based Systems, pp. 213–220 (2007)
17. Altschul, S.F., Gish, W., Miller, W., Meyers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. 215, 403–410 (1990)
18. Hunt, E., Atkinson, M.P., Irving, R.W.: Database indexing for large DNA and protein sequence collections. The VLDB J. 11, 256–271 (2002)
19. Fast Bit Counting Routines, `http://gurmeetsingh.wordpress.com/2008/08/05/fast-bit-counting-routines/`
20. PubChem Structure Search, `http://pubchem.ncbi.nlm.nih.gov/search/search.cgi`
21. JkDefrag Harddisk Defragger and Optimizer, `http://www.kessels.com/jkdefrag/`
22. Büttcher, S., Clarke, C.L.A.: Index compression is good, especially for random access. In: Proceedings of the 16th ACM Conference on Information and Knowledge Management, pp. 761–770 (2007)
23. Zhang, X., Berry, M.W., Raghavan, P.: Level search schemes for information filtering and retrieval. Info. Proc. & Mgt. 37, 313–334 (2001)
24. Baldi, P.: Chemoinformatics, drug design, and systems biology. Genome Info. 16, 281–285 (2005)