Blocked Inverted Indices for Exact Clustering of Large Chemical Spaces

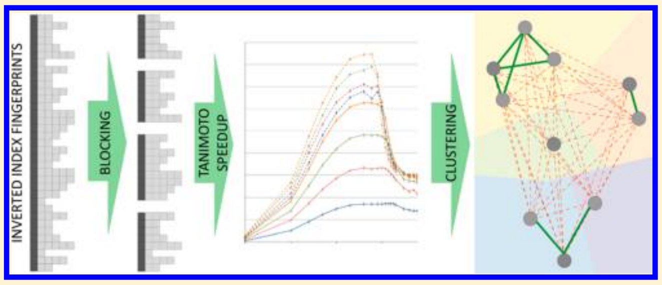
Philipp Thiel,*,[†] Lisa Sach-Peltason,[‡] Christian Ottmann,[§] and Oliver Kohlbacher[†]

[†]Applied Bioinformatics, Center for Bioinformatics, Quantitative Biology Center and Dept. of Computer Science, University of Tübingen, Sand 14, 72076 Tübingen, Germany

[‡]Pharma Research & Early Development Informatics, Data Science, F. Hoffmann-La Roche AG, Grenzacherstr. 124, CH-4070 Basel, Switzerland

[§]Laboratory of Chemical Biology and Institute of Complex Molecular Systems, Department of Biomedical Engineering, Technische Universiteit Eindhoven, Den Dolech 2, 5612 AZ Eindhoven, The Netherlands

S *Supporting Information*



ABSTRACT: The calculation of pairwise compound similarities based on fingerprints is one of the fundamental tasks in chemoinformatics. Methods for efficient calculation of compound similarities are of the utmost importance for various applications like similarity searching or library clustering. With the increasing size of public compound databases, exact clustering of these databases is desirable, but often computationally prohibitively expensive. We present an optimized inverted index algorithm for the calculation of all pairwise similarities on 2D fingerprints of a given data set. In contrast to other algorithms, it neither requires GPU computing nor yields a stochastic approximation of the clustering. The algorithm has been designed to work well with multicore architectures and shows excellent parallel speedup. As an application example of this algorithm, we implemented a deterministic clustering application, which has been designed to decompose virtual libraries comprising tens of millions of compounds in a short time on current hardware. Our results show that our implementation achieves more than 400 million Tanimoto similarity calculations per second on a common desktop CPU. Deterministic clustering of the available chemical space thus can be done on modern multicore machines within a few days.

INTRODUCTION

Calculating similarities between molecules is a fundamental task in chemoinformatics. The process of molecular similarity calculation comprises two steps, first the representation of molecules *in silico* and second the application of a similarity measure to pairs of molecules. Both steps have been researched extensively.[1] Similarity computations can be applied broadly, for example in compound library design, database search, virtual screening, and compound selection.[2−5] A common challenge in these applications is the huge and continuously growing number of compounds to be processed. This number ranges from small libraries of up to 10^5 compounds in academic set-ups to all commercially available compounds ($\sim 2 \times 10^7$ compounds). Recent research has led to sophisticated methods to calculate compound similarities, but these methods have mainly been focused on speeding up similarity searching.[6−8] Clustering is a versatile data mining technique able to partition a set of items. It is commonly used in chemoinformatics to create groups of similar molecules within a given set of compounds.[1] Various methodologies have been applied to cluster chemical data sets, and they can be grouped in hierarchic and nonhierarchic methods. Due to their space and runtime complexity, the use of hierarchical methods has so far been limited on compound data sets with less than a million molecules.[9] When larger compound libraries have to be clustered, nonhierarchic methods like *k*-means or variations thereof and the Jarvis-Patrick approach are used.[10−12] The *k*-means approach has several disadvantages like being nondeterministic or the need to specify the number of clusters as a parameter. Thus, mostly the Jarvis-Patrick method is employed to cluster libraries with millions of compounds. However, the algorithm has quadratic runtime complexity and thus heuristics are often used instead for large data sets.[13]

2D Fingerprint Similarity. A commonly used technique to express pairwise similarities between molecules is to compare their substructural compositions. The latter is usually encoded as a 2D fingerprint, which is a binary array where a distinct position indicates the absence (0-bit) or presence (1-bit) of a substructural element of the corresponding molecule. These elements can be explicit and predefined structural patterns (e.g., structural keys) or they can directly be calculated from every molecule within the data set as higher order features where the entirety of generated features forms a flexible virtual fingerprint (e.g., radial or path-based fingerprints).[14−16]

Received: March 10, 2014
Published: August 19, 2014

To compare the fingerprints of two molecules, two important parameters can be calculated from these fingerprints: (1) the number of 1-bits of every fingerprint itself, which is computationally easy and has to be done only once; (2) the number of shared 1-bits of a fingerprint pair to be compared, which is used as an estimate for their substructural overlap. This parameter is individual for every fingerprint pair and computationally quite demanding. Based on these simple parameters, various related similarity and dissimilarity coefficients have been defined.[17] One of the most intensively studied and frequently used similarity measures in chemoinformatics is the Jaccard[18] or Tanimoto[19] coefficient, which is defined as

$$S_{\tan} = \frac{c}{a + b - c}$$

where a is the number of 1-bits in fingerprint **A**, b is the number of 1-bits in fingerprint **B**, and c is the number of shared 1-bits of **A** and **B**. As already mentioned, the computational challenge of all similarity and dissimilarity coefficients based on these parameters is the calculation of the number of shared 1-bits, c. In set notation, the number of shared 1-bits of two fingerprints **A** and **B** is

$$c = |A \cap B|$$

Thus, the calculation can be split up into a binary AND operation of **A** and **B** followed by counting the 1-bits in the resulting array. Recently published solutions for chemoinformatic applications tackle this problem in two different ways. First, by using machine dependent instructions on the hardware level or second by sophisticated algorithms, which are based on so-called inverted index data structures.

Hardware-Accelerated Methods. The problem of counting the number of 1-bits in an array is well-known and often referred to as the population count (popcount) of an array. Its solution is of such high importance that recent instruction set extensions of modern CPUs (for example SSE4.2) as well as modern graphics processing units introduced this operation as single-cycle instructions on the hardware level. Haque and colleagues recently described a thoroughly implemented popcount based method and a combination with cache-efficient strategies, which show impressive performances.[20]

Inverted Index Methods. The inverted index data structure (iiDS) arose from the field of information retrieval and is used as a technique to encode text documents for subsequent similarity calculations.[21] Its relationship to tasks in the field of chemoinformatics has recently been described by Nasr and colleagues.[6] An inverted index algorithm to accelerate LINGO similarity calculations has also been described by Kristensen and colleagues.[22,23] Inverted indices store only the molecule features that are actually present, that is the indices of the 1-bits of the corresponding fingerprint. Thereby, an iiDS can take advantage of the sparseness of fingerprints by reducing the memory needed to store fingerprints and enables efficient calculation of shared feature counts.

In the present work, we describe an optimized algorithm for fast Tanimoto similarity calculation using inverted index data structures. We analyze the properties of this algorithm with a focus on parallel execution on modern multiprocessor machines and compare its performance to state-of-the-art similarity calculation implementations. As an application example for the developed similarity calculation method, an exact approach for clustering of large compound libraries is presented. The described clustering workflow is capable of processing libraries comprising tens of millions of compounds and permits deterministic clustering of all commercially available compounds in acceptable time on regular CPUs. The algorithms for fast similarity calculation and library clustering are implemented as a part of the open-source software framework BALL (Biochemical Algorithms Library).[24,25]

■ MATERIALS AND METHODS

Blocked Inverted Index Algorithm. A major aim was to create a pure algorithmic solution for this problem without hardware level instructions or specialized hardware. Thus, we used the concept of iiDS for our similarity calculation method. On the basis of the latter, we developed an improved and flexible algorithm for all pairwise similarity calculations. The algorithm takes as input a set of molecules encoded as 2D fingerprints. Internally, fingerprints are represented as feature lists. The key idea of our algorithm is to split the input molecules into equally sized blocks (Figure 1a) where the block

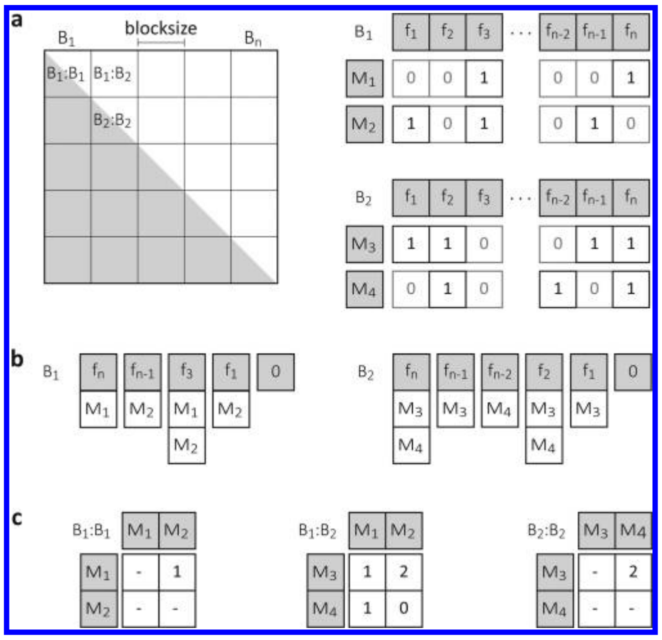


Figure 1. Blocked inverted index algorithm. (a) Left part: a blocked similarity matrix with a blocks size of two (i.e., every block contains two fingerprints). Right part: blocks B1 and B2 containing molecules M1−M4 represented as feature lists. (b) Blocks B1 and B2 transformed into inverted index data structures. (c) Processing of unordered block pairs from B1 and B2 to calculate the shared feature counts between the contained molecule pairs.

size is critical to the algorithm's performance as shown below. An iiDS is created for every block of molecules (Figure 1b), which in principle is a transposition of rows (molecule indices) and columns (feature indices) for a set of fingerprints. In more detail, an iiDS for a block **B** is a list of inverted indices, each corresponding to a unique feature of all fingerprints within **B**. An inverted index itself is a tuple of a unique index ID and a list storing the indices of all molecules within **B**, which possess the corresponding feature. In addition to the iiDS, a block stores the total number of features (1-bits) for every molecule.

On the basis of these data structures, the pairwise molecule similarities are calculated by processing every unordered pair of blocks including the processing of every block with itself

(Figure 1c). When processing a block pair, the algorithm iterates their iiDS and calculates the shared feature counts of all interblock fingerprint pairs, which are stored in the shared feature counts matrix. In its final state, this matrix holds the total shared feature counts of all fingerprint pairs between the compared iiDS. Iterating these fingerprint pairs allows the calculation of corresponding Tanimoto values. By processing all ordered iiDS pairs, the similarity matrix is incrementally constructed.

Clustering the Available Chemical Space. The clustering workflow described in the following is shown in Figure 2.

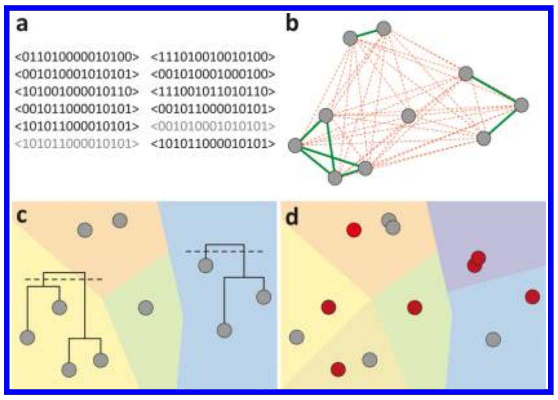


Figure 2. Clustering workflow. (a) Duplicate removal. (b) Calculation of all pairwise Tanimoto similarities and construction of a similarity network by applying a similarity threshold in order to retrieve induced connected components. (c) Application of hierarchical clustering on connected components. (d) Remapping of fingerprint duplicates onto clusters and medoid calculation.

The first step makes use of a nonuniqueness property of binary fingerprints. This is a consequence of the feature generation from substructural patterns, which normally do not encompass the entire molecule. Based on this property, the first step identifies and removes duplicate fingerprints to reduce the total number of similarity calculations in the subsequent steps (Figure 2a). The removed compounds are stored and finally remapped onto the clusters containing their representative fingerprint (Figure 2d).

The second step is a coarse grouping of the unique fingerprint set based on the concept of connected components (Figure 2b) in the similarity graph $G(V, E)$. In this graph, a vertex represents a compound and the set of edges is defined via the similarities between pairs of compounds. An edge (u, v) is added to E iff the similarity between the corresponding fingerprints exceeds a given threshold value. To construct E, all pairwise similarities are calculated using the blocked inverted index algorithm as described above. The resulting similarity graph decomposes into a set of connected components that correspond to a high-level clustering of the compound library.

The third step is a hierarchical agglomerative clustering of those connected components exceeding a predefined size (Figure 2c). As hierarchical clustering methodology, we have implemented average linkage for three reasons. First, the method has been shown to perform well for clustering of chemical databases.[26] Second, average linkage does not require cluster centroids and is thus compatible with our blocked inverted index algorithm. Third, for hierarchical agglomerative clustering, efficient algorithms have been described by Murtagh[27,28] on the basis of so-called reciprocal nearest neighbors (RNN). RNNs are cluster pairs that are mutual nearest neighbors and have the property that they can be merged immediately to create a new cluster. Murtagh described two RNN-based algorithms: (1) an iterative parallel approach where a nearest neighbor is calculated for every element with subsequent determination and merging of all RNN pairs at once. (2) The construction of a chain of subsequent nearest neighbors. At a certain point the chain ends up with an RNN pair, which has to be merged, and the algorithm continues from the last but two chain links. In our workflow, all pairwise similarities are calculated during similarity graph construction; thus we decided to keep the nearest neighbor information and start the hierarchical clustering using the parallel RNN version until the entire similarity matrix for the remaining clusters fits into memory. At this point, we calculate the similarity matrix for the remaining clusters and switch to the nearest neighbor chain algorithm to finish the clustering. For each hierarchically clustered connected component, the method of Kelley et al.[29] is used to automatically select a level for cluster assignment. Finally, we calculate the cluster medoids and remap the fingerprint duplicates that were removed in the first step (Figure 2d).

Implementation. The software is implemented in C++ and integrated in the open-source software framework BALL (version 1.4.1).[25] For the unique fingerprints filter, the feature lists are hashed using the collate class of the C++ standard library and duplicates are detected on the basis of the generated hash values. To calculate the connected components, we used the incremental connected components implementation of the BOOST graph library and the blocked similarity calculations have been parallelized using the BOOST thread library (versions: Intel system 1.55.0, AMD system 1.46.1). We used the C++ compiler from the GNU Compiler Collection (versions: Intel system 4.4.6, AMD system 4.4.7). The following relevant compiler flags were used: -O3 -fPIC -std=c++0x.

Software and Hardware. To benchmark the performance of our blocked inverted index algorithm, we compared it to the similarity search tool (simsearch) from the freely available and state-of-the-art chemoinformatics software package *chemfp* (version 1.1).[30] chemfp implements among others the popcount based method for fast Tanimoto calculation, which was introduced by Haque and colleagues.[20] chemfp was installed as described by the distributor, and the GNU OpenMP (GOMP, version 4.4.6) library for parallelization was used on the target platform. Cachegrind from the Valgrind profiling framework (version 3.6.0) was used to analyze software cache interactions. For performance evaluation of all methods, we used a current desktop computer equipped with an Intel Core i7-3770 processor (4 cores, 3.4 GHz) and 16 GB of main memory. For clustering of the ZINC all purchasable subset we used a computer server equipped with 4 AMD Opteron 6274 processors (64 cores, 2.2 GHz) and 512 GB of main memory.

Data Sets. To analyze the blocked inverted index implementation and for performance benchmarks, we used a subset comprising 100 000 molecules, which was randomly selected from the all purchasable subset of the ZINC database (ZINC version 12, accessed June 14, 2014).[31] Two different fingerprint types were calculated using the chemoinformatics software package RDKit (version 2013-09-01).[32] As representative for a sparse type, we used Morgan fingerprints (MorganFP), and for a dense type we used Layered fingerprints (LayeredFP). For both types, default RDKit parameters were used except for fingerprint length. Here, we created three data sets (1024, 2048, and 4096 bit length) for each type. Quantitative information for the fingerprints is listed in Table S1. For chemfp evaluation, bit-string fingerprints were converted into the FPS fingerprint format of chemfp using its tool sdf2fps.[33]As a compound library for our clustering workflow, we used the full ZINC all purchasable subset comprising 17 833 934 compounds at that time (ZINC version 12, accessed July 17, 2012). The library was prepared using Pipeline Pilot and OpenEye.[34,35] Salts were removed, and canonical SMILES were calculated. Extended-connectivity fingerprints with a maximum radius of four bonds were calculated using ChemAxon's JChem Base.[36]

■ RESULTS

The first part of this section analyzes the block size dependence of the blocked inverted index method and its consequences for parallelization. Second, we compare our blocked inverted index similarity method to the popcount based similarity search implementation from chemfp with a focus on the influence of fingerprint length and density on the methods performance. Finally, we analyze and process the ZINC all purchasable subset representing the available chemical space as an application example of our methods.

Block Size Dependency. The block size parameter defines the number of molecules for which a common iiDS is constructed. This parameter has significant impact on the performance of our algorithm. Increasing the block size mainly affects three important aspects of the method: (1) the number of pairwise block comparisons decreases, (2) the size of a single iiDS increases, and (3) the size of the shared feature counts matrix grows quadratic with the block size. The memory consumption of these data structures for 100 000 MorganFPs at varying block sizes is shown in Figure S1. It shows that the shared feature counts matrix is the dominating data structure for block sizes above 1800.

To analyze the block size impact on the performance of the blocked inverted index algorithm, we did similarity searches using 100 000 MorganFPs at varying block sizes and a Tanimoto threshold of 0.8. As a performance measure, we use the number of Tanimoto calculations per second (Tps). The results for single threaded execution are shown in Figure 3a. It also visualizes the corresponding miss rates of the L3 cache simulated by valgrind. Starting with a block size of one that corresponds to a single iiDS for every input molecule, the performance of our algorithm is with 2.6×10^6 Tps at its minimum and can be regarded as an efficient implementation of a naïve Tanimoto approach, which improves memory demands and prevents comparison of 0-bits. Increasing the block size leads to a hyperbolic performance increase, reaching its maximum at a block size of ~400. The throughput remains constant up to a block size of ~1800. Further increase leads to an asymptotic performance loss.

This performance characteristic can be explained by the algorithm's interaction with the L3 cache of the underlying CPU (Figure 3a). A small block size obviously leads to a high L3 miss rate limiting the Tanimoto throughput. Increasing the block size leads to decreasing miss rates and reaches optimal cache usage at block sizes above 400. Increasing the block size above 1800 again leads to increasing L3 miss rates and lowers the performance of the method. Blocking inverted indices thus can be exploited to achieve optimal CPU cache usage by compact fingerprint representation with low memory consumption. In the case of sparse fingerprints like MorganFPs, the average memory consumption per molecule ranges from 98 to 112 bytes per molecule (Table S1) and reduces the memory needs in comparison to bit vector fingerprints by more than 90%. This size reduction enables efficient shared feature count calculation for hundreds of fingerprint pairs at once.

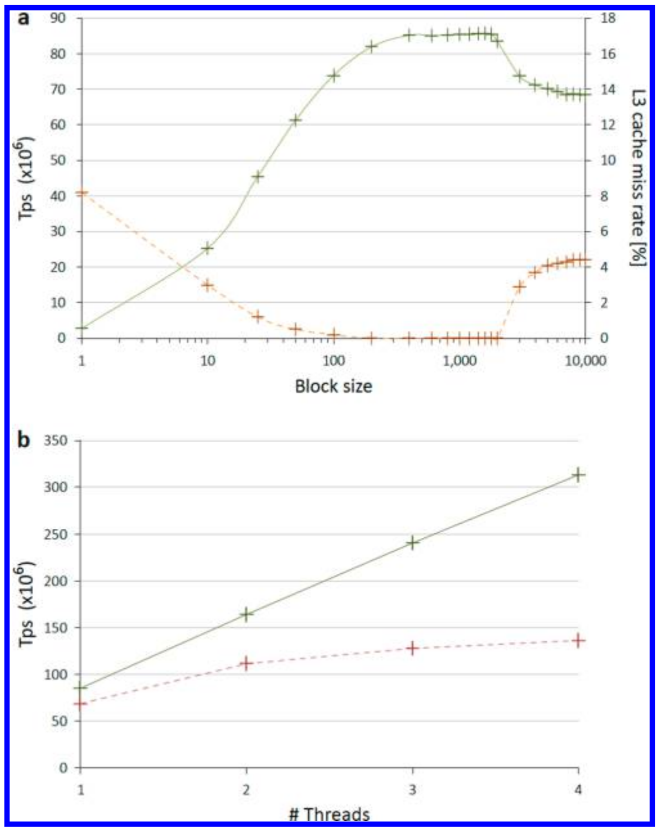


Figure 3. Blocked inverted index performance. (a) Single threaded performance for varying block sizes (solid line, primary vertical axis). L3 cache miss rates (dashed line, secondary vertical axis). (b) Scaling of blocked inverted index algorithm with the number of parallel threads. Performance measurements are shown for a block size of 600 (solid line) and a block size of 10 000 (dashed line).

An optimal choice of the block size is especially important if the algorithm is executed in parallel using multiple threads. Figures 3b and S2 illustrate the scaling of the blocked inverted index performance for one up to four threads at two different block sizes. A block size of 600 (green marks) that lies in the optimal range between 400 and 1800 scales linear with the number of threads. In contrast, at a block size of 10 000 the performance gain by use of multiple threads is nearly abolished. This observation is of the utmost importance for similarity matrix construction of large data sets because efficient parallel execution is necessary to achieve feasible run times.

Blocked Inverted Index Performance Comparison. To evaluate the performance of our blocked inverted index algorithm (block size = 600), we performed threshold searches for MorganFPs and LayeredFPs folded to different lengths (1024, 2048, and 4096 bits) at a Tanimoto threshold of 0.8. As a benchmark method, we used the popcount based similarity search implementation from chemfp also applied at a Tanimoto threshold of 0.8. The results are shown in Figure 4. It is noteworthy that both methods provide an exact problem solution by performing exhaustive similarity calculation of all fingerprint pairs and do not approximate the result.

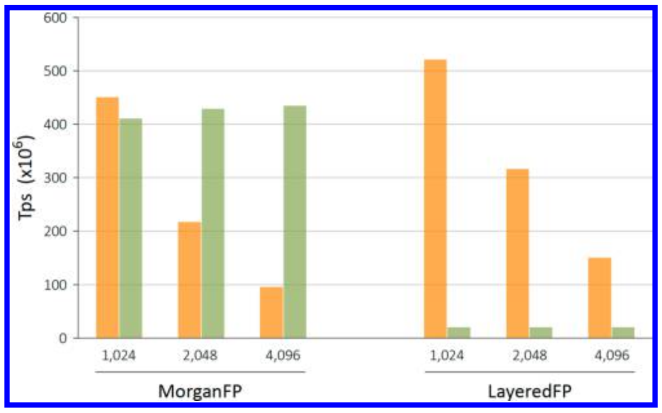


Figure 4. Performance results of blocked inverted index (green bars) and popcount based chemfp similarity search (orange bars). We compared two representative fingerprint types folded to different lengths.

For sparse fingerprints (MorganFP), the blocked inverted index algorithm calculates over 400×10^6 Tps for all lengths. For 1024 bit fingerprints, it is slightly slower than chemfp, but it is considerably faster for the longer fingerprint lengths. This is particularly remarkable because our method is a purely algorithmic solution for this problem. For dense fingerprints (LayeredFP), our method performs $\sim 20 \times 10^6$ Tps for all lengths and is significantly slower than chemfp. These results point out two important characteristics of the compared methods: (1) the performance of inverted index algorithms is rather independent of the fingerprint size, whereas the performance of popcount based algorithms decreases linearly with fingerprint length. (2) The performance of inverted index algorithms is output-sensitive with respect to the final shared feature counts whereas the performance of popcount based algorithms is independent thereof. The total numbers of shared features for the benchmark experiments are listed in Table S1.

Clustering the Available Chemical Space. For this experiment, we used the complete ZINC all purchasable subset encoded as extended-connectivity fingerprints. The unique fingerprints filter reduced the input library down to 12 976 486 fingerprints. Details on the accuracy of hashing binary fingerprints are provided in the Supporting Information. The time to perform hash-based redundancy reduction on the fingerprint level took ~5 min. In an intermediate step, we calculated all connected component decompositions in a Tanimoto range from 0.52 to 1.0 (step width = 0.2) to choose appropriate thresholds for similarity graph generation and connected component size exclusion for the hierarchical clustering (Figure 5). The graph shows a similar behavior as previously reported by Lepp and colleagues for three different, small compound data sets.[2] They also recognized a single huge connected component that decomposed in a Tanimoto range of 0.5 to 0.8 and the second largest component being an order of magnitude smaller than the largest one.

On the basis of these results, we decided to set the threshold for similarity network generation to a Tanimoto value of 0.76 and the size exclusion threshold for hierarchical clustering of a connected component to 1000. That is, components with a size below 1000 molecules were not directed to hierarchical clustering. Using these parameters, we executed the workflow on the Opteron compute server using all cores and 100 GB of main memory, which took 64 h to complete.

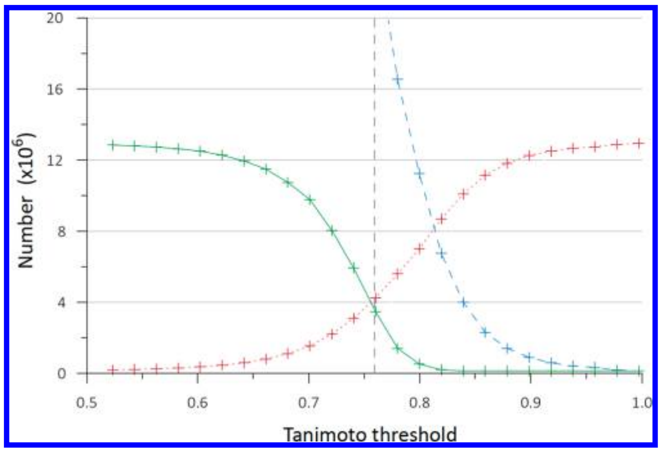


Figure 5. Similarity network analysis of the ZINC all purchasable subset. The plot shows the number of edges exceeding the similarity threshold (dashed line), the number of connected components (dotted line), and the size of the largest connected component (solid line). The selected Tanimoto threshold used for clustering is highlighted as a dashed vertical line.

■ DISCUSSION

In the present study, we describe an efficient algorithm for all pairwise Tanimoto similarity calculation using binary fingerprints on large compound libraries. It is based on a data structure known as inverted index, which is a memory efficient representation of binary fingerprints enabling fast calculation of shared feature counts between hundreds of fingerprint pairs at once. We have modified this data structure by introducing a block size parameter to divide the input molecules into small and equally sized inverted index blocks in order to maximize the number of similarity calculations per second. The block size parameter enables optimal CPU cache usage by reducing cache miss rates. Thus, this parameter allows choosing an optimal trade-off between inverted index size and speed of similarity calculations. Especially for parallel execution, an optimally chosen block size guarantees linear scaling with the number of physical CPU cores. The algorithm is thus perfectly suited for the current trend of increasing the number of CPU cores. Recently introduced methods to calculate fingerprint similarities use popcount instruction set extensions on the hardware level or specialized hardware like GPUs. In contrast, the presented blocked inverted index algorithm is general and can be run on any architecture. Comparison to popcount based methods demonstrated that our blocked inverted index method performs comparably well for sparse fingerprints and significantly better if their length exceeds 1024 bits. For dense fingerprints, popcount based methods are significantly faster than our method, independent of fingerprint length.

Based on this algorithm, we implemented an exact, deterministic clustering workflow that is able to process large compound libraries. The workflow combines different techniques to decompose an input library in three main steps: unique fingerprint filtering, graph-based connected components decomposition, and hierarchical clustering. To demonstrate the ability of our clustering workflow to process large data sets, we clustered the available chemical space comprising more than 17 million compounds on a compute server, which took 2.5 days of calculation time.

In conclusion, we are of the opinion that our blocked inverted index algorithm is a useful extension of the set of available methods for high performance similarity calculation using 2D binary fingerprints. Especially for the current trend to increase the number of CPU cores, our results are of great value because an optimal blocking of inverted index fingerprints is essential to exploit parallel resources. The recent and ongoing progress in algorithmic as well as hardware development thus could pave the way to process considerably larger data sets than ever before. We have exemplified this with our clustering workflow, which hopefully will serve as a useful tool for clustering of large chemical spaces on standard hardware.

■ ASSOCIATED CONTENT

Ⓢ Supporting Information

Further information on fingerprint hashing, fingerprint statistics (Table S1), blocked inverted index memory requirements (Figure S1), and parallelization (Figure S2). This material is available free of charge via the Internet at http://pubs.acs.org. Additionally, the described algorithms are contained in the developmental version of BALL (LGPL), which is available at https://bitbucket.org/ball/ball. The tools *FingerprintSimilaritySearch* and *FingerprintSimilarityClustering* provide access to the algorithms. The MorganFP and LayeredFP data sets used to benchmark the methods can be downloaded from http://abi.inf.uni-tuebingen.de/People/thiel/materials/.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: thiel@informatik.uni-tuebingen.de.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

We gratefully acknowledge the use of software from ChemAxon and OpenEye Scientific Software, Inc. This work was supported by the DFG (Grant OT 414/1-2; P.T. and C.O.), the BMBF (MoSGrid, 01IG09006C), and the EU (SCI-BUS, Contract No. 283481; P.T. and O.K.), AstraZeneca, Bayer CropScience, Bayer HealthCare, Boehringer Ingelheim, Merck KGaA and the Max Planck Society (P.T. and C.O.).

■ ABBREVIATIONS

iiDS, inverted index data structure; popcount, population count; RNN, reciprocal nearest neighbor; MorganFP, RDKit Morgan fingerprint; LayeredFP, RDKit Layered fingerprint; Tps, Tanimoto calculations per second

■ REFERENCES

(1) Leach, A. R.; Gillet, V. J. *An Introduction to Chemoinformatics*, 1st ed.; Springer: New York, 2007.

(2) Lepp, Z.; Huang, C.; Okada, T. Finding Key Members in Compound Libraries by Analyzing Networks of Molecules Assembled by Structural Similarity. *J. Chem. Inf. Model.* **2009**, *49*, 2429−2443.

(3) Ripphausen, P.; Nisius, B.; Bajorath, J. State-of-the-Art in Ligand-Based Virtual Screening. *Drug Discovery Today* **2011**, *16*, 372−376.

(4) Baldi, P.; Hirschberg, D. S.; Nasr, R. J. Speeding up Chemical Database Searches Using a Proximity Filter Based on the Logical Exclusive Or. *J. Chem. Inf. Model.* **2008**, *48*, 1367−1378.

(5) Vainio, M. J.; Kogej, T.; Raubacher, F. Automated Recycling of Chemistry for Virtual Screening and Library Design. *J. Chem. Inf. Model.* **2012**, *52*, 1777−1786.

(6) Nasr, R. J.; Vernica, R.; Li, C.; Baldi, P. Speeding Up Chemical Searches Using the Inverted Index: The Convergence of Chemoinformatics and Text Search Methods. *J. Chem. Inf. Model.* **2012**, *52*, 891−900.

(7) Smellie, A. Compressed Binary Bit Trees: A New Data Structure for Accelerating Database Searching. *J. Chem. Inf. Model.* **2009**, *49*, 257−262.

(8) Cao, Y.; Jiang, T.; Girke, T. Accelerated Similarity Searching and Clustering of Large Compound Sets by Geometric Embedding and Locality Sensitive Hashing. *Bioinformatics* **2010**, *26*, 953−959.

(9) Varin, T.; Bureau, R.; Mueller, C.; Willett, P. Clustering Files of Chemical Structures Using the Székely-Rizzo Generalization of Ward's Method. *J. Mol. Graphics Modell.* **2009**, *28*, 187−195.

(10) Jarvis, R. A.; Patrick, E. A. Clustering Using a Similarity Measure Based on Shared Near Neighbors. *IEEE Trans. Comput.* **1973**, *C-22*, 1025−1034.

(11) Forgy, E. W. Cluster Analysis of Multivariate Data: Efficiency vs Interpretability of Classifications. *Biometrics* **1965**, *21*, 768−769.

(12) Böcker, A.; Derksen, S.; Schmidt, E.; Teckentrup, A.; Schneider, G. A Hierarchical Clustering Approach for Large Compound Libraries. *J. Chem. Inf. Model.* **2005**, *45*, 807−815.

(13) ChemAxon User's Guide 6.1.3. http://www.chemaxon.com/jchem/doc/user/Jarp.html.

(14) Daylight Theory Manual. http://www.daylight.com/dayhtml/doc/theory/.

(15) Bender, A.; Mussa, H. Y.; Glen, R. C.; Reiling, S. Molecular Similarity Searching Using Atom Environments, Information-Based Feature Selection, and a Naïve Bayesian Classifier. *J. Chem. Inf. Comput. Sci.* **2003**, *44*, 170−178.

(16) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50*, 742−754.

(17) Holliday, J. D.; Hu, C.-Y.; Willett, P. Grouping of Coefficients for the Calculation of Inter-Molecular Similarity and Dissimilarity Using 2D Fragment Bit-Strings. *Comb. Chem. High Throughput Screening* **2002**, *5*, 155−166.

(18) Jaccard, P. Étude Comparative de La Distribution Florale Dans Une Portion Des Alpes et Des Jura. *Bull. Soc. Vaudoise Sci. Nat.* **1901**, *37*, 547−579.

(19) Tanimoto, T. T. *IBM Internal Report*; IBM: Armonk, NY, 1957.

(20) Haque, I. S.; Pande, V. S.; Walters, W. P. Anatomy of High-Performance 2D Similarity Calculations. *J. Chem. Inf. Model.* **2011**, *51*, 2345−2351.

(21) Zobel, J.; Moffat, A. Inverted Files for Text Search Engines. *ACM Comput. Surv.* **2006**, *38*, 6−es.

(22) Vidal, D.; Thormann, M.; Pons, M. LINGO, an Efficient Holographic Text Based Method to Calculate Biophysical Properties and Intermolecular Similarities. *J. Chem. Inf. Model.* **2005**, *45*, 386−393.

(23) Kristensen, T. G.; Nielsen, J.; Pedersen, C. N. S. Using Inverted Indices for Accelerating LINGO Calculations. *J. Chem. Inf. Model.* **2011**, *51*, 597−600.

(24) Kohlbacher, O.; Lenhof, H. P. BALL−Rapid Software Prototyping in Computational Molecular Biology. Biochemicals Algorithms Library. *Bioinformatics* **2000**, *16*, 815−824.

(25) Hildebrandt, A.; Dehof, A. K.; Rurainski, A.; Bertsch, A.; Schumann, M.; Toussaint, N. C.; Moll, A.; Stöckel, D.; Nickels, S.; Mueller, S. C.; Lenhof, H.-P.; Kohlbacher, O. BALL−Biochemical Algorithms Library 1.3. *BMC Bioinformatics* **2010**, *11*, 531.

(26) Downs, G. M.; Willett, P.; Fisanick, W. Similarity Searching and Clustering of Chemical-Structure Databases Using Molecular Property Data. *J. Chem. Inf. Model.* **1994**, *34*, 1094−1102.

(27) Murtagh, F. A Survey of Recent Advances in Hierarchical Clustering Algorithms. *Comput. J.* **1983**, *26*, 354−359.

(28) Murtagh, F. *Multidimensional Clustering Algorithms*; Compstat, L., Ed.; Physica-Verlag: Würzburg-Wien, 1984.

(29) Kelley, L. A.; Gardner, S. P.; Sutcliffe, M. J. An Automated Approach for Clustering an Ensemble of NMR-Derived Protein Structures into Conformationally Related Subfamilies. *Protein Eng.* **1996**, *9*, 1063−1065.

(30) Dalke, A. *chemfp*, version 1.1; Dalke Scientific; Sweden, 2013. http://chemfp.com/.

(31) Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; Coleman, R. G. ZINC: A Free Tool to Discover Chemistry for Biology. *J. Chem. Inf. Model.* **2012**, *52*, 1757−1768.

(32) Landrum, G. RDKit: Cheminformatics and Machine Learning Software. http://www.rdkit.org/ (version 2013-09-01).

(33) Dalke, A. The FPS Fingerprint Format and Chemfp Toolkit. *J. Cheminform.* **2013**, *5*, P36.

(34) *Pipeline Pilot 8.0*; Accelrys: San Diego, CA. http://accelrys.

(35) *OEChem*, version 1.5.4, OpenEye Scientific Software, Inc., Santa Fe, NM. www.eyesopen.com, 2008.

(36) *JChem 5.8.0*; ChemAxon: Budapest, Hungary, 2011. http://www.chemaxon.com.