Chemical Compounds with Path Frequency Using Multi-Core Technology

Kun-Ming Yu[1], Yi-Yan Chang[2], Jiayi Zhou[3], Chun-Yuan Huang[1], Whei-meih Chang[1], Chun-Yuan Lin[4], Chuan Yi Tang[5]

[1]Department of Bioinformatics, Chung Hua University
[2]Department of Information Management, Chung Hua University
[3]Institute of Engineering and Science, Chung Hua University
[4]Department of Computer Science and Information Engineering, Chang Gung University
[5]Department of Computer Science, National Tsing Hua University

yu@chu.edu.tw, {frank38, jyzhou}@pdlab.csie.chu.edu.tw, {chunyuan.huang, wmchang}@chu.edu.tw, cyulin@mail.cgu.edu.tw, cytang@cs.nthu.edu.tw

Abstract. Drug design is the approach of finding drugs by design using computational tools. When designing a new drug, the structure of the drug molecule can be modeled by classification of potential chemical compounds. Kernel Methods have been successfully used in classifying chemical compounds, within which the most popular one is Support Vector Machine (SVM). In order to classify the characteristics of chemical compounds, methods such as frequency of labeled paths have been proposed to map compounds into feature vectors. In this study, we analyze the path frequencies computed from chemical compounds, and reconstruct all possible compounds that share the same path frequency with the original ones, but differ in their molecular structures. Since the computation time for reconstructing such compounds increase greatly along with the size increase of the compounds, we propose an efficient algorithm based on multi-core processing technology. We report here that our algorithm can infer chemical compounds from path frequency while effectively reduce computation time and obtained high speed up.

Keywords: Chemical compound, feature space, Multi-Core Processing, Branch-and-Bound, OpenMP

1 Introduction

In recent years, many researchers have worked on the drug design problem in order to develop new drugs based on computation methods. When designing a new drug, the structure of the drug molecule can be modeled by classifying candidate chemical compounds using Kernel Methods [4, 5, 6, 7], within which the most popular one is Support Vector Machine (SVM) [10]. Kernel method is a type of pattern analysis, the task of which is to discover the relationships, such as clusters, rankings, classifications, in the data (such as sequences, vectors, sets of points, images, etc). Kernel methods approach the problem by first mapping the data into a high-

dimensional feature space. Recently, it has also been applied to the classification of chemical compounds [4, 5, 6, 7]. In these approaches, chemical compounds are mapped to feature vectors and then SVMs [9, 10] are employed to learn the rules for classifying these feature vectors. Several mapping methods for feature vectors have been proposed; among them, the mapping of feature vectors based on the *frequency of labeled paths* [6, 7] or the *frequency of small fragments in chemical compounds* [4, 5] are widely used.

In kernel methods, an object in the input space can be mapped into a point (or feature vector) in a space called feature space. Through a suitable function $\emptyset$, a given point y in the feature space can be mapped back into an object in the input space. Such object is called pre-image. The problem exists when mapping a given y in feature space back into an object in the input space such that $y=\emptyset(x)$ is satisfied, as x may not exist.

In [1], a feature vector g is a multiple set of strings of labels with length at most K which represents path frequency. Given a feature vector g, they considered the problem of finding a vertex-labeled graph G that attains a one-to-one correspondence between g and the set of sequences of labels along all paths of length at most K in G.

In previous works [1, 2], a graph can be inferred from the numbers of occurrences of vertex-labeled paths. In [1], they showed that this problem can be solved in polynomial time of the size of an output graph if graphs are trees of bounded degree and the lengths of given paths are bounded, by a constant, whereas this problem is strongly NP-hard even for planar graphs of bounded degree.

In this study, we have taken into account the situation when chemical compounds become increasingly complex, the computation time required to infer pre-images from the feature vectors of these compounds increase at a much faster rate. We resort to parallel computing, in which the computation tasks are assigned to multiple cores appropriately to reduce the overall computation time. We extend the algorithms in [3], and therefore the modified algorithms can support multi-core processing technology.

The rest of this paper is organized as follows. Section 2 introduces the background about problem and definition. Next we describe our proposed algorithms in section 3. In section 4, we show the experimental result. Finally we conclude this paper in section 5.

2 Related Work

For classification of the characteristics of chemical compounds to work, chemical compounds are often mapped into feature vectors. Several methods for converting chemical compounds into feature vectors have been proposed. Among them, methods such as frequency of labeled paths [6, 7] or frequency of small fragments [4, 5] are popular. Recently, the pre-image methods have been proposed. In [4], pre-images were found in a general setting by using Kernel Principal Component Analysis and regression. In [8], stochastic search algorithm is used to find pre-images for graphs. However, these pre-image methods are not derived from a computational viewpoint.

In [4], the obtained results and performance of the algorithm was unclear because it was applied only to a few similar cases. Other related pre-image studies include inferring a tree from walks in [12], as well as inferring by graphic reconstruction [13].

In [3], chemical structures are modeled as trees or tree-like structures. They extend algorithms in [1, 2] so that constraints on valences of atoms are taken into account. They proposed an algorithm, *Branch-and-Bound Chemical compound Inference from Path Frequency* (BB-CIPF), which can infer tree from related chemical structures. BB-CIPF works within a few or a few tens of seconds for inferring moderate size of chemical compounds (e.g., the number of carbon atoms are less than 20) with tree or tree-like structures, and can be modified for inferring more general classes of chemical compounds and/or for feature vectors based on frequency of small fragments.

In BB-CIPF, given a tree T^{cur} to be inferred to a target tree T^{target}, T^{cur} is first inserted into a node n to become T^{next}. If the feature vector f^{next} of T^{next} does not comply with the feature vector f^{target} of T^{target}, the T^{next} will be discarded and then the T^{cur} will be re-inserted into another node and be compared to T^{target}.

The advantage of BB-CIPF algorithm is to effectively reduce the computation time, as it terminates the computation process immediately and displays the results once it obtains a solution; this also means that there is only one solution [3]. For example, if there are three objects, *a*, *b* and *c*, which all correspond to the same feature vectors *v*. Through BB-CIPF algorithm, only one of the objects *a*, *b*, *c* can be inferred from *v*, so the inferred solution is not necessarily be the most useful one in practice. Therefore, how to produce all possible compounds that are mapped back from the same feature vector but differ in their molecular structures is an important issue in the problem. Moreover, when a compound structure is more complex, it will require more computation time for inference of its solutions.

Parallel computing is a suitable technique in shortening the inference procedure. Parallel computing is a form of computation in which several calculations are carried out simultaneously [11], operated on the principle that large problems can often be divided into smaller ones, and then solved concurrently to provide the solution in a shorter time. While clusters, Massive parallel processing (MPP), and Grids use multiple computers to work on the same task, multi-core and multi-processor computers employ multiple processing elements to work on the same task.

A multi-core processor (or chip-level multiprocessor) combines two or more independent cores (normally a CPU) into a single package that consisted of a single integrated circuit. A dual-core processor contains two cores, and a quad-core processor contains four cores. A multi-core microprocessor implements several processing units in a single physical package. In general, programming is required to orchestrate processes in several cores in order to solve problems.

The OpenMP (Open Multi-Processing) standard allows programmers to take advantage of the new shared-memory, multiprocessor programming systems from vendors like Compaq, Sun, HP, and SGI. Aimed at the researcher working with C/C++ or Fortran programming languages, OpenMP explains both what this standard is and how to use it to create software that takes full advantage of parallel computing. OpenMP support Sun compiler, GNU compiler and Intel compiler.

In this paper, we extend the inference algorithm [3] to obtaining all possible compounds that are mapped back from the same feature vector but differ in their molecular structures. We used the Branch-and-Bound concept to derive the trees or tree-like structures of chemical compounds. Our algorithm is committed to obtain all possible compounds that can be inferred from the same feature vector but differ in their molecular structures. We develop our algorithm based upon the algorithm in [3] so that the computation process will not terminate on the first obtained solution, but will continue to search for all possible solutions. However, in order to output more chemical compounds, it also means that the algorithm will consume more computation time. Therefore, we also propose adopting the multi-core computing technology to reduce the computation time in our proposed algorithm. We hope that by providing more thorough and practical solutions to the inference problem, we can improve on the development of drug design.

3 Multi-Core Chemical Compound Inference from Path Frequency (MC-CIPF)

In the previous section, we have described that when a compound structure is more complex, it will require more computation time for inference of its solutions. That is to say, if the feature vector v in feature space has been mapped from a compound *c* thought a function ∅, and we want to find *c'* where *c'*= ∅(*v*). If a compound is more complex in structure, its feature vector in feature space is also more complex, and it will require substantially more computation time to map back to *c'* from *v*. Therefore, in this paper, we divide computation tasks into several smaller tasks and distribute these tasks appropriately among several processing cores for computation. We propose the Multi-Core Chemical Compound Inference from Path Frequency (MC-CIPF) to obtain all possible compounds.

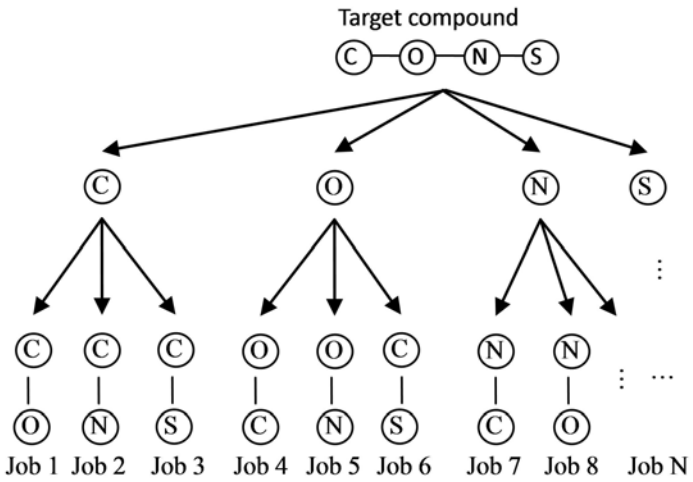

Fig. 1. Each job is initiated based on the atoms that existed in the target compound.

In the first step of MC-CIPF, the algorithm loads into the master core a target compound for inference of all other chemical compounds that share the same feature vector. The master core employs the Breadth-First-Search (BFS) algorithm to analyze the target compound and obtain its path frequency for distributing jobs later. Each job is initiated based on the atoms that exist in the target compound (Fig. 1). However, H atoms are not included in this step.

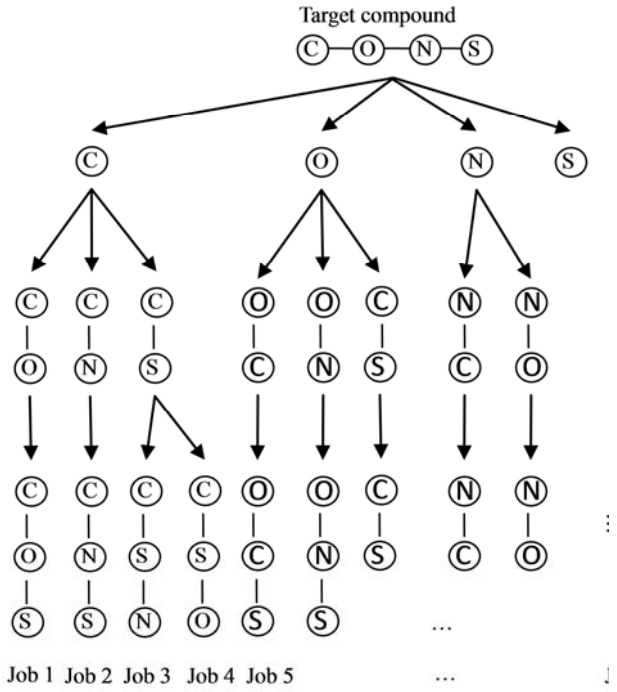

Fig. 2. An example of balancing the load in each core in MC-CIPF.

Each job requires different amount of time for computation, and a more complex one will require more time. For example, if there are four cores C_1, C_2, C_3 and C_4, the master core will analyze the target compound, initiate four jobs T_1, T_2, T_3, T_4, and distribute them among four corresponding cores for execution. If T_1, T_2 and T_3 have completed their jobs while T_4 is still in process, T_1, T_2 and T_3 cores will be in idle as there are no more jobs to allocate to these cores. Therefore, we balance the overall computation loads by increasing the number of jobs that can be allocated by the master core and reducing the computation time demanded for each job. The scenario is depicted in Fig. 2.

Each core applies the Depth-First-Search (DFS) algorithm to insert an atom into a candidate compound. After inserted an atom, the candidate compound will be compared with target compound, and if the feature vector of candidate compound has the same structure as parts of target compound structure, the inserted atom will be

kept; otherwise, the inserted atom will be dropped, and the algorithm continues on applying DFS to insert the next atom in queue into the candidate compound. If the resulted structure of the candidate compounds is in line with the target structure, it is output as one of the solution. The algorithm iterates until all cores have completed all candidate compounds in their queues.

```
Procedure BFS (T^targe)
  Let T^queue be a queue that stores all candidate compounds;
  for all a ∈ all atoms exist in T^target do
    Let T^temp be a temporary compound
    T^temp ← ∅ ;
    Insert a into T^temp;
    if T^temp ∈ T^target then
      Add T^temp to T^queue;
    else
      continue (examine the next atom in T^target);
    end
  end
  while T^queue is not empty do
    for each core compute a compound from T^queue per time do
      Compute feature vector f^temp_i from T^temp_i in T^queue;
      if MC-CIPF(T^temp_i, f^temp_i, T^target, f^target)=false then
        output "no solution";
      end
    end

Procedure MC-CIPF(T^temp_i, f^temp_i, T^target, f^target)
  if f^temp_i = f^target then output T^temp_i;
    popup T^temp_i from T^temp;
    return true;
  else
    return false;
  for all a ∈ all atoms exist in T^target do
    L ← ∅ ;
    if { L(u)|u ∈V(T^temp)} ∪ {a} ⊈ atomset(f^target) then
      continue;
    for all w ∈ V(T^temp) do
      Let T^next be a tree gotten by connecting new leaf u with
label a to w by bond b;
      if w does not satisfy the valence constraint then
        continue;
      Compute f^next from T^next and f^temp;
      if MC-CIPF(T^next, f^next, T^target, f^target)=true then
        return true;
      end
    end
  return false;
```

4 Experimental Results

For verifying the effectiveness of MC-CIPF, we implemented the proposed algorithm and compared the performance when using single core, dual core and quad core for computation. The simulation environment is built by using a personal computer equipped with Intel Core 2 Quad Q6600 CPU and 4 GB RAM and installed with the operating system of Windows Vista with Service Pack 1. MC-CIPF was implemented using C language and experimental datasets are retrieved from KEGG LIGAND Database.

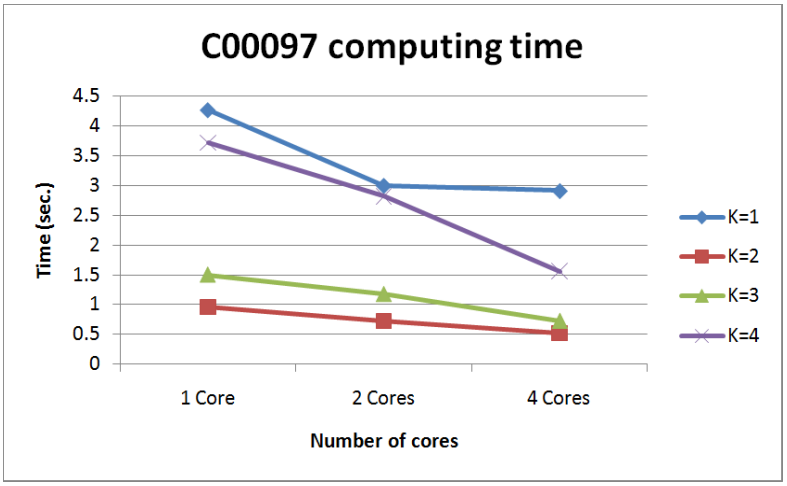

Fig. 3(a). Computing time of C00097.

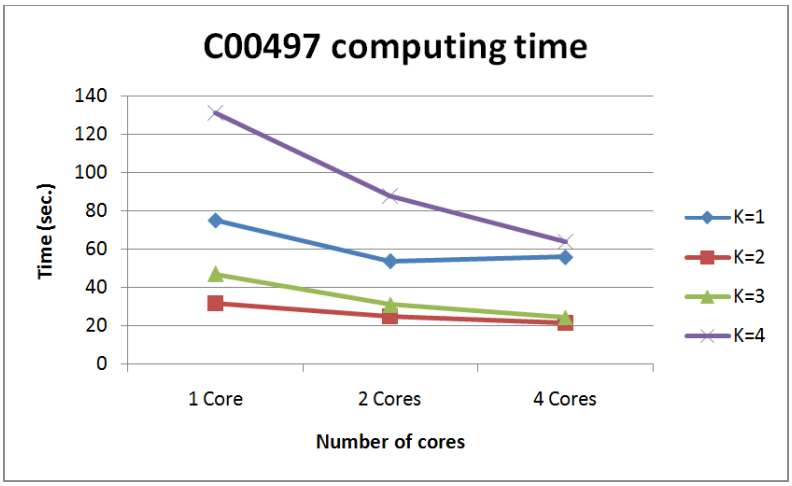

Fig. 3(b). Computing time of C00497.

In the experiment, we randomly chosen 5 chemical compounds (C00097, C00497, C11109, C14601, and C15987; the number of atoms of compound size with hydrogen are 14, 15, 16, 15 and 19, respectively) from KEGG LIGAND Database and examined them with K = 1, 2, 3, 4, where K is the length of sequence label of feature vectors in MC-CIPF. Larger K means more constraints for target compound, which leads to less variation for its molecular structure. Fig. 3(a)-(e) are the computing time for each chemical compound. In each case, we have found that the computing time was reduced as the number of cores was increased. For example, in Fig. 3(c), when K was equal to 4, the computing time was reduced from 11.9337 second with 1 core to 5.542821 second with 4 cores.

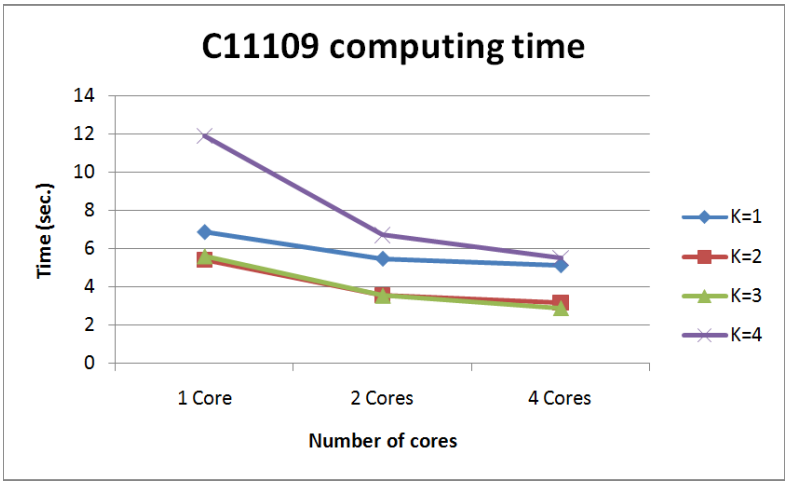

Fig. 3(c). Computing time of C11109.

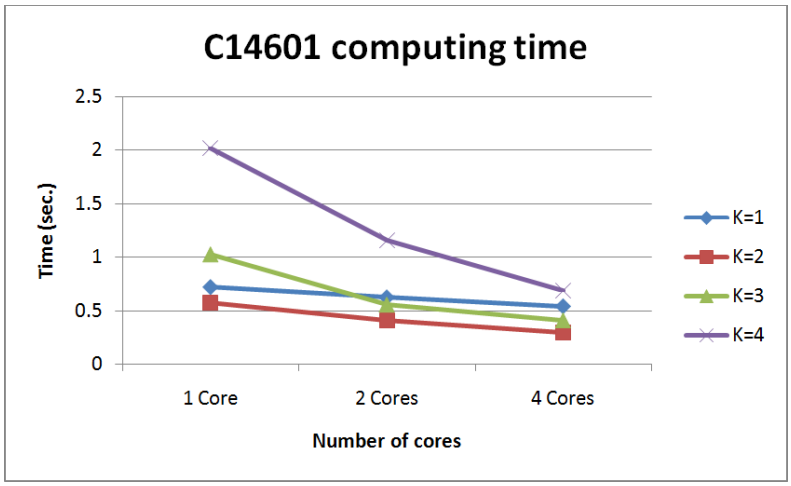

Fig. 3(d). Computing time of C14601.

When the *K* value increases, the constraints of the feature vectors increase accordingly. As a result, MC-CIPF spends less computing time in searching for the combinations of target compound, as there is less permitted variations to compute for. However, the path frequency will be longer, so MC-CIPF needs to spend more execution time in re-computing path frequency. Consequently, the shortest computing time occurs when *K* is equal to 2 in the experiment, since the number of constraints has not increased too much and the length of path frequency is not too long. Details of the computing time are shown in Table 1.

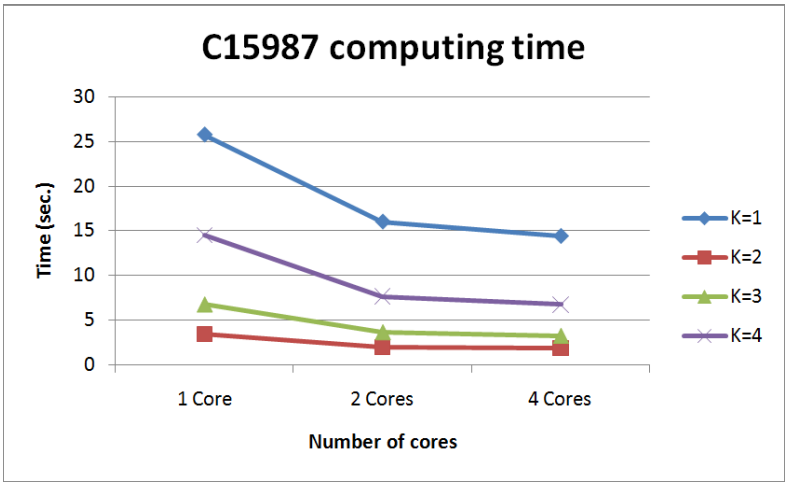

Fig. 3(e). Computing time of C15987.

Table 1. Computing time of MC-CIPF for various chemical compounds.

Instances	Cores detail		CPU time (sec.)			
			K=1	K=2	K=3	K=4
C00097	1 Core		4.27	0.96	1.49	3.71
	2 Cores	Core 1	2.99	0.26	1.17	2.81
		Core 2	3.00	0.72	1.17	2.81
		Max	3.00	0.72	1.17	2.81
	4 Cores	Core 1	2.91	0.20	0.72	1.56
		Core 2	2.67	0.36	0.46	0.98
		Core 3	0.77	0.35	0.50	0.97
		Core 4	2.91	0.51	0.72	1.56
		Max	2.91	0.51	0.72	1.56
C00497	1 Core		75.44	31.95	47.23	131.42
	2 Cores	Core 1	28.82	8.78	31.26	35.33
		Core 2	54.10	25.12	31.27	88.09
		Max	54.10	25.12	31.27	88.09
	4 Cores	Core 1	13.76	3.93	24.43	11.05
		Core 2	31.23	7.77	12.25	28.88
		Core 3	12.47	7.88	12.26	30.69
		Core 4	56.14	21.67	24.43	64.10
		Max	56.14	21.67	24.43	64.10
C11109	1 Core		6.88	5.42	5.63	11.93
	2 Cores	Core 1	5.47	3.59	3.55	6.73
		Core 2	5.48	3.59	3.56	6.73
		Max	5.48	3.59	3.561	6.73
	4 Cores	Core 1	5.13	3.18	2.90	5.53
		Core 2	1.83	2.06	1.91	3.53
		Core 3	2.48	2.05	1.18	1.93
		Core 4	5.13	3.18	2.90	5.54
		Max	5.13	3.18	2.90	5.54
C14601	1 Core		0.72	0.57	1.02	2.02
	2 Cores	Core 1	0.62	0.40	0.51	0.83
		Core 2	0.62	0.40	0.55	1.16
		Total	0.62	0.40	0.55	1.16
	4 Cores	Core 1	0.54	0.27	0.40	0.37
		Core 2	0.13	0.23	0.38	0.55
		Core 3	0.21	0.25	0.40	0.68

		Core 4	0.54	0.29	0.35	0.58
		Max	0.54	0.29	0.40	0.68
C15987	1 Core		25.73	3.49	6.77	14.52
	2 Cores	Core 1	16.02	2.00	0.35	4.95
		Core 2	11.43	2.00	3.67	7.67
		Max	16.02	2.00	3.67	7.67
	4 Cores	Core 1	14.41	1.87	1.28	2.03
		Core 2	5.78	1.48	1.65	3.16
		Core 3	12.85	1.87	3.25	6.76
		Core 4	1.42	0.72	0.55	0.68
		Max	14.41	1.87	3.25	6.76

More importantly, we want to compare the speedup ratios of MC-CIPF with respect to the core number used in the experiments. Fig. 4(a)-(e) are the speedup ratios of C00097, C00497, C11109, C14601, and C15987. In these figures, the speedup ratios are increased from 1 core to 4 cores, with the best speedup ratio close to 3 (Fig. 4(c)). Interestingly, when the *K* value is increased, the speedup ratio is raised accordingly.

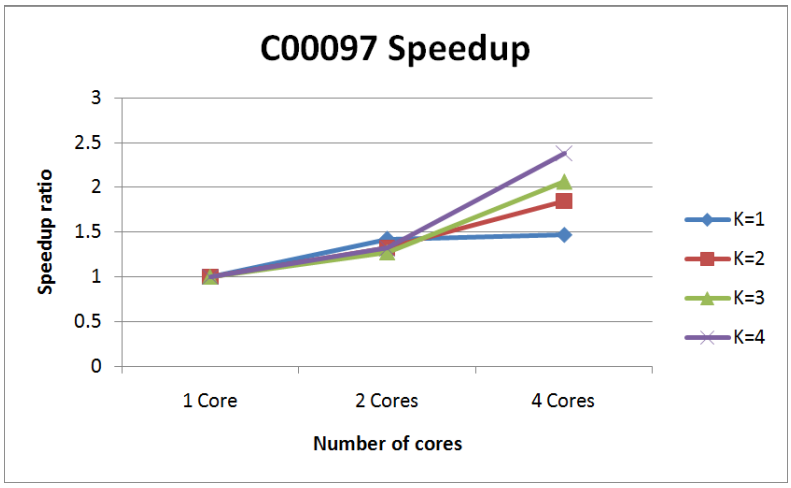

Fig. 4(a). Speedup ratio of C00097

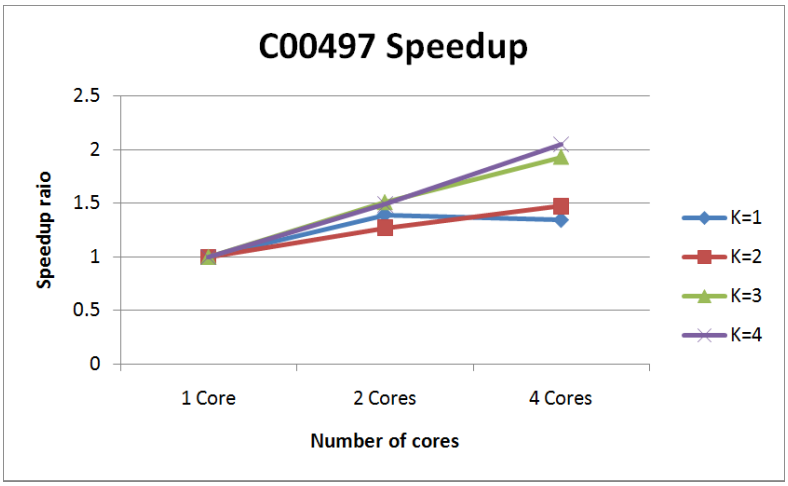

Fig. 4(b). Speedup ratio of C00497.

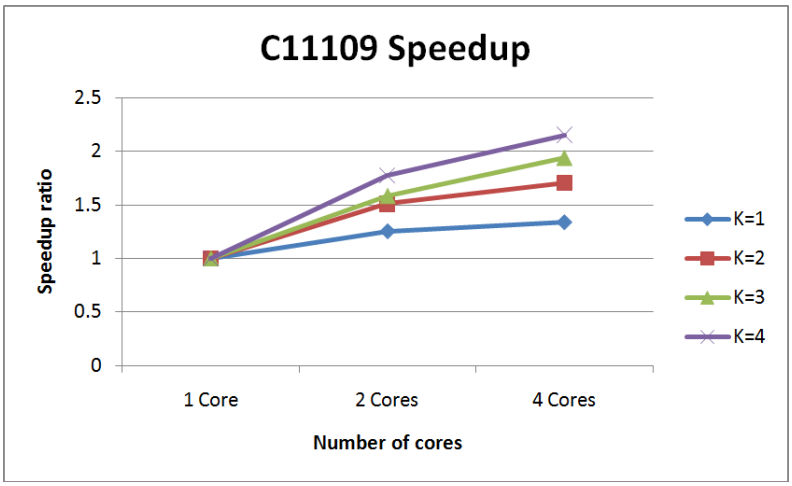

Fig. 4(c). Speedup ratio of C11109.

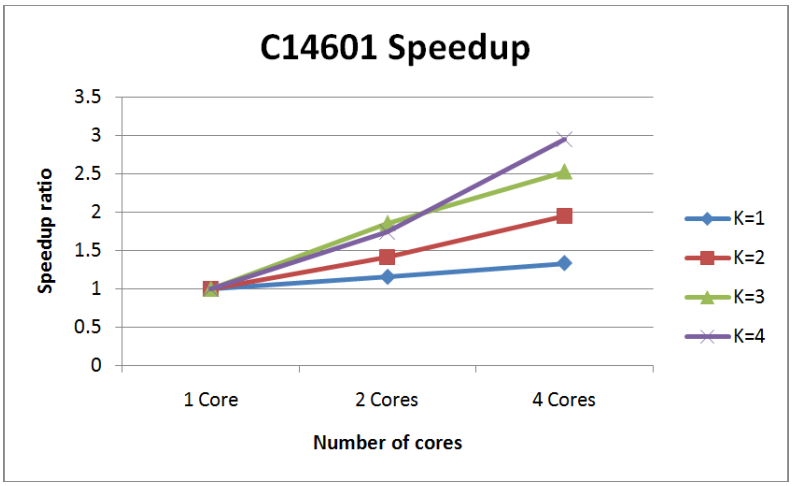

Fig. 4(d). Speedup ratio of C14601.

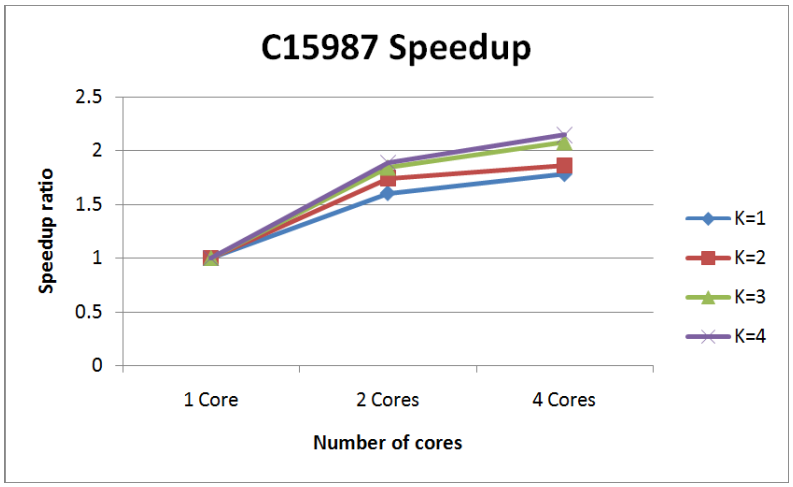

Fig. 4(e). Speedup ratio of C15987.

5 Conclusions

In this research, we proposed a multi-core algorithm for solving Chemical Compound Inference from Path Frequency problem. We adopted the Branch-and-Bound concept to evolve the tree-like structures of chemical compounds in the paper. The experimental results show that our algorithm can practically reduce computing

time, with the best speedup ratio close to 3 folds while using 4 cores in the experiment. Therefore, our proposed algorithm can infer chemical compounds from path frequency effectively and reduce computation time by employing the multi-core technology.

References

1. Tatsuya, A., Daiji, F.: Inferring a graph from path frequency, In Proc. 16th Symp. Combinatorial Pattern Matching. Lecture Notes in Computer Science, Springer. 2527, 371--392 (2005)
2. Tatsuya, A., Daiji, F.: On inference of a chemical structure from path frequency. Proc. 2005 International Joint Conference of InCoB, AASBi, and KSBI, pp. 96--100 (2005).
3. Tatsuya, A., Daiji, F.: Inferring a Chemical Structure from a Feature Vector Based on Frequency of Labeled Paths and Small Fragments. APBC, pp. 165--174 (2007)
4. Byvatov, E., Fechner, U., Sadowski, J., Schneider, G.: Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. Journal of Chemical Information and Computer Sciences. 43, 1882--1889 (2003)
5. Deshpande, M., Kuramochi, M., Wale, N., Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds. IEEE Trans. Knowledge and Data Engineering. 17, 1036--1050 (2005)
6. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In Proc. 20th Int. Conf. Machine Learning, pp. 321--328 (2003)
7. Mahé, P., Ueda, N., Tatsuya, A., Perret, J-L., Vert, J-P.: Graph kernels for molecular structure-activity relationship analysis with support vector machines. Journal of Chemical Information and Modeling. 45, 939--951 (2005)
8. Bakir, G.H., Zien, A., Tsuda, K.: Learning to find graph pre-images. In Proc. The 26th DAGM Symposium. Lecture Notes in Computer Science, Springer. 3175, 253--261 (2004)
9. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning. 20, 273--297 (1995)
10. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge Univ. Press (2000)
11. Almasi, G.S. and A. Gottlieb. Highly Parallel Computing. Benjamin-Cummings publishers, Redwood City, CA. (1989)
12. Maruyama, O., Miyano, S.: Inferring a tree from walks. Theoretical Computer Science. 161, 289--300 (1996)
13. Lauri, J., Scapellato, R.: Topics in Graph Automorphisms and Reconstruction. Cambridge Univ. Press (2003)