社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006−MPS−60 (13)
2006／6／26

グリッド環境下での分散型ワーカモデルを用いた Modified PrefixSpan 法の動的負荷分散方式

高木 允[†] 田村慶一[††] 北上 始[††]

アミノ酸配列からモチーフとなりうる頻出パターンを高速に抽出するために，我々はグリッド環境下で Modified PrefixSpan 法の並列化に関する研究を進めている．Modified PrefixSpan 法は 2 つの特徴を持っている．ひとつは並列化した際に極端な負荷の偏りが生じることであり，もうひとつはタスクの負荷を予め見積もることができないことである．本研究では，グリッド環境下での Modified PrefixSpan 法の並列処理において分散型ワーカモデルを適用した．さらに，マルチキャストとランダムスティール法を組み合わせた動的負荷分散方式である Cache-based Multicast Stealing (CMS) を提案する．分散型ワーカモデルは PC クラスタ数の増加に対応でき，CMS は PC クラスタ間の通信遅延によるオーバヘッドを削減することができる．

Dynamic Load Balancing Technique for Modified PrefixSpan on a Grid Environment with Distributed Worker Model

MAKOTO TAKAKI ,[†] KEIICHI TAMURA [††] and HAJIME KITAKAMI[††]

Motif is a characteristics pattern that is biologically meaningful in an amino acid sequence. In order to extract the frequent sequence patterns that can become a motif in amino acid sequences at high speed, we are working on developing the parallel processing of the Modified PrefixSpan method on a grid environment. The Modified PrefixSpan method has two characteristics: One is an extreme load imbalance and the other is the inability to estimate the load of the task. In this study, the distributed worker model is applied to the parallel processing of the Modified PrefixSpan method on a grid environment. Moreover, we propose Cache-based Multicast Stealing (CMS), which combines the multicast and Cache-based Random Stealing(CRS) technique. The distributed worker model has enough scalability to endure an increase in the number of PC clusters. CMS can reduce the overhead generated by the communication delay among the PC clusters.

1. Introduction

In the field of molecular biology, the focus has been on the extraction of motifs. A motif is assumed to be related to a function of proteins that have been preserved in the evolutionary process of an organism. In order to extract a motif, frequent sequence patterns with variable-length wildcard regions must be extracted from amino acid sequences.

In order to extract the frequent sequence patterns with fixed-length and variable-length wildcard regions, the Modified PrefixSpan method[1)] has been proposed. The Modified PrefixSpan method is an algorithm that extends the PrefixSpan method[2)]. The PrefixSpan method is a tree-projection-based frequent sequence pattern extraction algorithm. The algorithm of tree-projection-based frequent sequence pattern extraction has high parallelism. We have been working on developing the parallel processing of the Modified PrefixSpan method on a PC cluster[3)].

This paper presents the parallel Modified PrefixSpan method on a grid environment. The distributed worker model[4)] is applied to the parallel processing of the Modified PrefixSpan method on a grid environment. The distributed worker model has more scalability to endure the increase in the number of PC clusters than master-worker model

† 広島市立大学大学院情報科学研究科
Graduate School of Information Sciences, Hiroshima City University
†† 広島市立大学情報科学部
Faculty of Information Sciences, Hiroshima City Unievrsity

表 1 アミノ酸配列の例
Table 1 Example of Amino Acid Sequences

sid \ position	1	2	3	4	5	6	7	8
1	F	K	Y	A	K	W	L	
2	S	F	V	K	T	A		
3	A	L	H					
4	M	S	K	P	L			
5	F	S	K	F	L	M	A	W

such as the Hierarchical master-worker model[5] on a grid environment.

Moreover, we propose a new dynamic load-balancing technique, which is called *Cache-based Multicast Stealing* (CMS). CMS combines multicast stealing and *Cache-based Random Stealing*[6]. CMS can distribute the workloads among the PC clusters according to a very large load imbalance and minimize the communication overhead among the PC clusters.

We evaluated CMS on an artificial grid environment with DummyNet. In the performance evaluation, three PC clusters consisting of four PCs each were used. A dataset that includes motifs called Kunitz was used. We confirmed that CMS can minimize the effect of the communication delay and the load imbalance among the PC clusters.

The rest of the paper is organized as follows: Section 2 explains the Modified PrefixSpan method. Section 3 is an explanation of the parallel processing of the Modified PrefixSpan method on a grid environment, and Section 4 proposes the new dynamic load-balancing technique. Section 5 discusses the related work. Section 6 shows the performance evaluation. Section 7 is the conclusion.

2. Modified PrefixSpan

This section explains the problem definition, the basic algorithm of the Modified PrefixSpan method and an example of frequent sequence pattern extraction processing.

2.1 Problem Definition

Let Σ = {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y } be a set of all letters in amino acid sequences. Sequence s is denoted as "$a_1 a_2 \cdots a_m$", where a_j is a letter, i.e., $a_j \in \Sigma$, and $a_j = s[j]$ for $1 \le j \le m$. Sequence database S is a set of tuples $\langle sid, s_{sid} \rangle$, where sid is a sequence identifier and s_{sid} is a sequence. Table 1 shows an example of the amino acid sequences.

Users specify three parameters. The parameters are the minimum support (*min_sup*), the maximum length of wildcard regions (*max_wc*) and the maximum number of errors (ε_{max}). A pattern which satisfies the user specified parameters is extracted as a frequent sequence pattern.

A k-length sequence pattern is denoted as $\langle pat^k \rangle$ = $\langle A_1\text{-}x(i_1,j_1)\text{-}A_2\text{-}x(i_2,j_2)\text{-} \cdots \text{-}x(i_{k-1},j_{k-1})\text{-}A_k \rangle$. Symbol "$A_j$" is called a character element. Symbol "-" means that the next element is continued. Representation $x(i_n, j_n)$ denotes variable-length wildcard regions, where $0 \le i_n \le max\_wc$ and $0 \le j_n - i_n \le \varepsilon_{max}$. A wildcard region indicates an arbitrary letter string. If $j_n = i_n$, $x(i_n, j_n)$ means the fixed-length wildcard regions.

The support of $\langle pat^k \rangle$ in sequence database S is the number of tuples containing $\langle pat^k \rangle$. If the support of frequent sequence pattern $\langle pat^k \rangle$ is cnt, this frequent sequence pattern is denoted as "$\langle pat^k \rangle$: cnt". Hereafter, a k-length frequent sequence pattern is called a k-frequent sequence pattern.

For example, if users specify *min_sup*=3, *max_wc*=3 and ε_{max}=3, the Modified PrefixSpan method can extract pattern $\langle F\text{-}x(2,5)\text{-}A \rangle$ in sequence database S given in Table 1. $\langle F\text{-}x(2,5)\text{-}A \rangle$ includes patterns that are $\langle F\text{-}x(2)\text{-}A \rangle$, $\langle F\text{-}x(3)\text{-}A \rangle$, $\langle F\text{-}x(4)\text{-}A \rangle$ and $\langle F\text{-}x(5)\text{-}A \rangle$. If a wildcard is represented as "*," the sequence s_1 includes "F**A," s_2 includes "F***A," and s_5 includes "F*****A." The number of tuples that include pattern $\langle F\text{-}x(2,5)\text{-}A \rangle$ is more than *min_sup*. Therefore, pattern $\langle F\text{-}x(2,5)\text{-}A \rangle$ is extracted as a frequent sequence pattern.

2.2 Algorithm

The basic algorithm of the Modified PrefixSpan method is pattern growth, such as the k-frequent sequence pattern, into (k+1)-frequent sequence patterns. In order to grow a pattern from $\langle pat^k \rangle$ into $\langle pat^{k+1} \rangle$, a projected database is required. The projected database of $\langle pat^k \rangle$ is denoted as PDB($\langle pat^k \rangle$) = $\{(i,j) \mid s_i \in S$, the next position of the rightmost character of $\langle pat^k \rangle$ is the value of j, where $1 \le j \le \| s_i \|\}$. Pattern $\langle pat^{k+1} \rangle$ can be generated using PDB($\langle pat^k \rangle$) by only investigating the next position of the rightmost character of $\langle pat^k \rangle$.

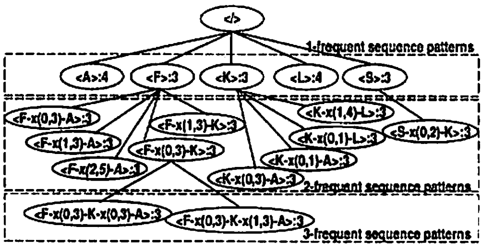

図 1 パターン抽出例
Fig. 1 Extraction of Patterns

To extract the frequent sequence patterns with variable-length wildcard regions, the Modified PrefixSpan method requires a scope database that keeps the (k+1)-length candidate patterns including wildcard regions between $\langle pat^k \rangle$ and α. The scope database of $\langle pat^k \rangle$ is denoted as $\mathrm{SDB}(\langle pat^k \rangle, [r, r + \varepsilon_{max}]) = \{(i, wc, s_i[j_{new}]) \mid (i, j) \in \mathrm{PDB}\ (\langle pat^k \rangle), wc \in [r, r + \varepsilon_{max}], j_{new} = j + wc, 1 \leq j + wc \leq \| s_i \|, s_i \in S\}$, where $r \in [0, max\_wc]$.

The algorithm of the Modified PrefixSpan method is as follows.

(1) First, 1-frequent sequence patterns are extracted by scanning sequence database S . Then, PDB($\langle pat^1 \rangle$) are generated.

(2) SDB($\langle pat^k \rangle, [r, r + \varepsilon_{max}]$), where $k \geq 1$, and (k+1)-length candidate patterns are generated by using PDB($\langle pat^k \rangle$). SDB($\langle pat^k \rangle$, $[r,\ r+\varepsilon_{max}]$) has the position information of (k+1)-length candidate patterns, which is concatenated in the variable-lenght wildcard region, whose length is $[r,\ r + \varepsilon_{max}]$ and is concatenated in an alphabet α to the suffix of $\langle pat^k \rangle$. SDB which has the position information of the (k+1)-length pattern with α in the suffix, is denoted as SDB_α.

(3) For all candidate (k+1)-length patterns which are included in SDB_α, the support is counted by using SDB_α. If the support of candidate (k + 1)-length patterns is over min_sup, the candidate (k+1)-length patterns are the (k+1)-frequent sequence patterns $\langle pat^{k+1} \rangle$. Then, for each $\langle pat^{k+1} \rangle$, PDB($\langle pat^{k+1} \rangle$) is created by SDB_α.

(4) If no PDB($\langle pat^{k+1} \rangle$) is generated, the Modified PrefixSpan method terminates the process.

表 2 スコープデータベース (SDB(⟨ F⟩,[2,5]))
Table 2 Scope Database (SDB(⟨ F⟩,[2,5]))

sid \ ε	0	1	2	3($=\varepsilon_{max}$)
1	F**A	F***K	F****W	F*****L
2	F**T	F***A	-	-
3	-	-	-	-
4	-	-	-	-
5	F**F F**A	F***L F***W	F****M	F*****A

Otherwise, go to step (2).

2.3 Example

Let our running database be sequence database S given in Table 1 with min_sup=3, max_wc=3 and ε_{max}=3. Figure 1 shows all frequent sequence patterns. First, 1-frequent sequence patterns are extracted by scanning sequence database S. $\langle A \rangle, \langle F \rangle, \langle K \rangle, \langle L \rangle$, and $\langle S \rangle$ are extracted as 1-frequent sequence patterns (Figure 1). Let us consider extracting the 2-frequent sequence patterns from $\langle F \rangle$ when the number of wildcards is r, where r=2. Table 2 indicates the scope database of pattern $\langle F \rangle$, i.e., SDB($\langle F \rangle, [2, 2+3]$). Here, let us focus on SDB_A. In Table 2, the patterns that have "A" in suffix of $\langle F \rangle$ are in three tuples. The support count of $\langle F\text{-}x(2,5)\text{-}A \rangle$ is over min_sup(=3). Therefore, "$\langle F\text{-}x(2,5)\text{-}A \rangle$:3" is extracted as a 2-frequent sequence pattern.

3. Modified PrefixSpan on a Grid Environment

3.1 Distributed Worker Model

In this study, the distributed worker model[4] is applied to the parallel processing of the Modified PrefixSpan method on a grid environment. The master-worker model[7] and the distributed worker model are known as the parallelization model.

In the master-worker model, the master becomes the bottleneck if the number of PC clusters is increased. However, the distributed worker model has enough scalability to endure the increase in the number of PC clusters. In addition, each worker can communicate with other workers (PC clusters) directly. Bottlenecks involving specific PC clusters will rarely occur.

Each PC cluster has a global PC. The global PC communicates with other global PCs. The PC cluster consists of a global PC and multiple workers. In the PC cluster, the master-worker structure is con-

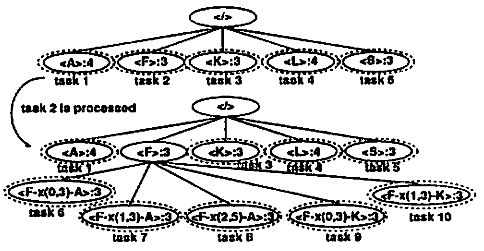

図 2 小粒度タスク
Fig. 2 Small-Grain Tasks

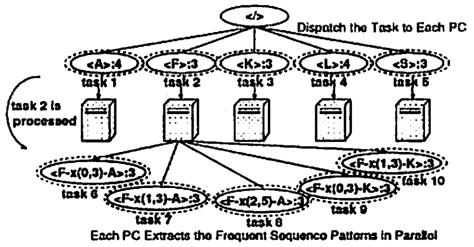

図 3 小粒度タスクの並列実行例
Fig. 3 Parallel Processing of Small-Grain Tasks

structed. The global PC is the master, and the other PCs are the workers.

3.2 Small-Grain Task

In the parallel processing of the Modified PrefixSpan method, the task is defined as a "small-grain task." Figure 2 is an illustration of an example of a small-grain task. Figure 3 illustrates an example of parallel processing of small-grain task. The characteristics of a small-grain task are as follows.

(1) The small-grain task consists of a k-frequent sequence pattern and its projected database.

(2) Processing a small-grain task is equal to the extraction of the $(k+1)$-frequent sequence patterns from a k-frequent sequence pattern (Figure 2).

(3) The small-grain task can be processed independently in each PC (Figure 3).

(4) Generated $(k+1)$-frequent sequence patterns are partial solutions. In addition, $(k+1)$-frequent sequence patterns are new small-grain tasks (Figure 2).

(5) The total number of frequent sequence patterns that will be extracted from a small-grain task and the total processing time cannot be estimated beforehand.

Frequent sequence pattern extraction processing is parallelized by distributing the small-grain tasks to each PC like Figure 3. The transfer of the load is easily because of the characteristics of the small-grain task.

4. Dynamic Load Balancing

4.1 Load Imbalance Problem on a Grid Environment

It is necessary to solve the very large load imbalance among the PC clusters. The load imbalance among the PCs is very large in the parallel processing of the Modified PrefixSpan method with the distributed worker model on a PC cluster[6]. Moreover, the load of the small-grain task cannot be estimated beforehand. Therefore, the load imbalance among the PC clusters is very large in the parallel processing of the Modified PrefixSpan method on a grid environment.

In previous study, we proposed a robust dynamic load-balancing technique called Cache-based Random Stealing (CRS)[6] for the parallel processing of the Modified PrefixSpan method on a PC cluster. CRS is a dynamic-load balancing technique that has a cache function with the Random Stealing (RS) technique. CRS can steal a task by priority from a PC with an extremely large work load.

However, the following problem exists when CRS is used on a grid environment. CRS does not consider communication delays among the PC clusters. CRS causes an increase in the communication overhead on a grid environment because a global PC might select other global PCs whose communication delay is maximal when sending the task request message.

4.2 Cache-based Multicast Stealing Schema

To address the above-mentioned problem of the load imbalance on a grid environment, we propose a dynamic load-balancing technique called CMS. CMS combines multicast and CRS. CMS can reduce the communication overhead more than CRS. An overview of CMS follows.

In CMS, as a PC cluster becomes idle, the global PC sends a task request message to all other global PCs first. The global PC caches the identifier of the donor PC cluster that sent the small-grain task to the source global PC earliest to CID. As the PC

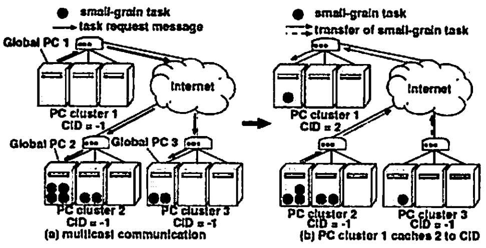

図 4 Cache-based Multicast Stealing
Fig. 4 Cache-based Multicast Stealing

cluster becomes idle again, the global PC looks up the CID. If the value of CID is valid, the global PC selects the PC cluster whose identifier is CID as a donor PC cluster and sends the task request message to the donor PC cluster. If the value of CID is invalid, the global PC sends the task request message to all other PC clusters again.

4.3 Procedure of CMS

Detailed procedure of CMS are given below. The following procedure indicates the process of the global PC. Each PC has the task pool to store the small-grain tasks. Figure 4 illustrates the procedure of CMS.

(1) Each global PC has a variable CID with a default value of -1. Each global PC extracts the 1-frequent sequence patterns by scanning the sequence database.

(2) The global PC distributes the small-grain tasks by using the round-robin technique. The global PC puts the 1-frequent sequence patterns to the task pool as the small-grain tasks.

(3) The small-grain tasks are executed inside the PC cluster. If there are no tasks inside the PC cluster, step (4) is followed.

(4) If CID is equal to -1, step (a) is followed. If $CID \geq 1$, step (b) is followed.

- (a) The global PC sends the task request message to all other global PC (Figure 4(a)). Step (5) is followed.
- (b) The global PC sends the task request message to the global PC which has CID as the identifier. Step (5) is followed.

(5) If the multicast was selected in step (4), step (a) is followed. If the cached ID was selected in step (4), step (b) is followed.

- (a) If the requested global PC received a small-grain task as the reply of the task request message, the global PC caches the ID of the source global PC that sent the small-grain task to the requested global PC earliest to CID. The global PC stores the received small-grain task to the task pool. Step (3) is followed. If no task can be received from other global PCs, step (6) is followed.
- (b) If the requested global PC received a small-grain task as the reply of task request message, the global PC stores the received small-grain task to the task pool. Step (3) is followed. If no task was received, the global PC changes CID to -1. Step (4) is followed.

(6) The global PC sends the termination signal to all PC inside the PC cluster and all other global PC.

CMS can make a certain global PC receives a task from the global PC whose communication delay is minimal and that has the tasks according to a change in the communication delay and load imbalance.

4.4 Example

Let us consider the example of CMS for Figure 4. In Figure 4, the focus is on PC cluster 1.

First, global PC 1 in PC cluster 1 sends the task request message to all other global PCs (Figure 4(a)) because of CID is equal to -1. Then, all other global PCs send a small-grain task to the requested global PC (Figure 4(b)). If global PC 1 receives the small-grain task from global PC 2 first, global PC 1 caches the identifier of global PC 2 to the CID. When global PC 1 has no task, global PC 1 sends the task request message to global PC 2 because the CID is equal to 2. If global PC 1 receives no task from global PC 2, global PC 1 changes the CID to -1.

5. Related Work

The hierarchical master-worker (HMW) model has been proposed[5)] as a parallelization model on a grid environment. The HMW model consists of a supervisor, the masters and the workers. There are multiple groups that consist of a master and multiple workers. The supervisor manages the multiple masters. The HMW model covers the branch and bound applications. In the HMW model, the speed of processing is increased by pruning unnecessary problems and synchronizing the temporary solution among the PC clusters. Therefore, the HMW model is suitable to branch and bound applications.

In the HMW model, each master communicates only with the supervisor. If the distance between the supervisor and the groups is large, the processing time is increased because of the communication overhead. The network topology is important problem in the HMW model.

On the other hand, in the distributed worker model, each PC cluster can communicate with all other PC clusters. Each PC cluster can select a PC cluster whose communication delay is minimal. The network topology is not a significant problem in the distributed worker model.

In the Modified PrefixSpan method, the load imbalance among the PC clusters is extreme. Active communication among the PC clusters is required. Thus, the distributed worker model is applied as the parallelization model in this study.

6. Performance Evaluation

In order to evaluate the effectiveness of CMS, CRS, Multicast Stealing (MS), Random Stealing (RS), and Nearest Stealing (NS) are evaluated. In RS, a global PC selects other global PCs randomly. In MS, a global PC selects all other global PCs when there are no tasks inside the PC cluster. In NS, a global PC select other global PCs based on the order that a user specified beforehand.

We evaluated CMS, CRS, RS, MS, and NS as the dynamic load balancing for the Modified PrefixSpan method on a grid environment. Three PC clusters that consist of four PCs are used. The DummyNet[8] was used in order to construct the artificial grid environment. The DummyNet can generate the communication delay artificially. Three PC clusters are connected with the DummyNet. Each PC was configured with a 2.8GHz Pentium4 processor with 1.0GB memory. The PCs were connected with a 1000 Mbit/sec Ethernet inside the PC cluster. Fedora Core 2 was used as the operating system. The socket and the MPI library were used. The dataset used in this evaluation was provided by PROSITE[9]. The dataset that includes the Kunitz motif has 70 data records (total length: 23,385 bytes).

The communication delay between PC cluster 1 and PC cluster 2 is 100[msec], between PC cluster 2 and PC cluster 3, 200[msec], and between PC cluster 1 and PC cluster 3, x [msec], where $x \in \{100, 200, \cdots, 500\}$. The parameters are as follows: $min\_sup = 35(50\%)$, $max\_wc = 5$ and $\varepsilon_{max} = 2$.

Figure 5 shows the processing time. Figure 6 shows the number of communications, and Figure 7 shows the amount of communication. The horizontal axis is the communication delay between PC cluster 1 and PC cluster 3. The vertical axis is the processing time (Figure 5), the number of communications (Figure 6) and the amount of communication (Figure 7).

In Figure 5, CMS can keep the processing time nearly flat. However, NS, RS, and MS cannot keep the processing time flatly as the communication delay increased. In NS, a global PC can select another PC cluster with a minimal communication delay. However, NS cannot select the PC cluster with an very large load. The number of times that the task request is missed increases. The same holds true for RS. Therefore, in NS and RS, the number of communications and the amount of communication are increased more than they are in other dynamic load-balancing techniques. This is shown in Figure 6 and Figure 7.

MS can reduce the number of communications, as shown in Figure 6. However, the amount of communication is increased in Figure 7. MS sends the task request message to all PC clusters every time. Unnecessary communication to the PC cluster that has no task increases. In CRS, the communication time is nearly flat. CRS can select a PC cluster with extreme loads. However, the random nature increases the number of communications.

In the evaluation with three PC clusters, CMS can keep the processing time nearly flat. Moreover, CMS can reduce the number of communications and the amount of communication by reduc-

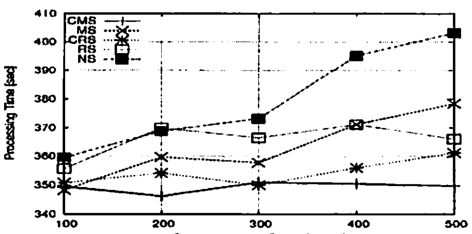

図 5 処理時間
Fig. 5 Processing Time

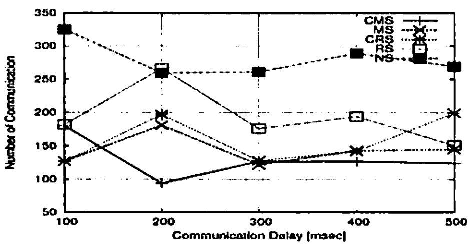

図 6 通信回数
Fig. 6 Number of Communication Events

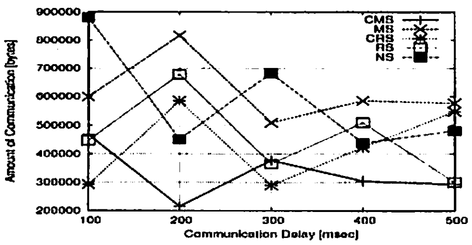

図 7 通信量
Fig. 7 Amount of Communication

ing unnecessary communications more than other techniques.

7. Conclusions

This paper presents the parallel processing of the Modified PrefixSpan method on a grid environment. The distributed worker model is applied to the parallel Modified PrefixSpan method on a grid environment. Moreover, we have proposed CMS. CMS can address the communication delay and very large load imbalance among the PC clusters. In this study, we have evaluated the proposed technique in an artificial grid environment. In order to prove the effectiveness of CMS, we plan to evaluate CMS in an actual grid environment.

Acknowledgments This work was made possible in part by the support of a Grant-in-Aid for Scientific Research(C)(2)(No.17500097) from the JSPS and a Hiroshima City University Grant for Special Academic Research (general research fund - code number 3106).

References

1) Shigetaka Tono, Hajime Kitakami, Keiichi Tamura, Yasuma Mori, and Susumu Kuroki. Efficiently Mining Sequence Patterns With Variable-Length Wildcard Regions Using An Extended Modified PrefixSpan Method. In *The 13th Annual International Conference on Intelligent Systems for Molecular Biology*, p. 147, 2005.
2) Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE Trans. Knowl. Data Eng.*, Vol.16, No.11, pp. 1424–1440, 2004.
3) Makoto TAKAKI, Keiichi TAMURA, Toshihide SUTOU, and Hajime KITAKAMI. Dynamic Load Balancing for Parallel Modified PrefixSpan. In *Proceedings of The 2004 International Conference on Parallel and Distributed Processing Techniques and Applications*, pp. 352–358. CSREA Press, 2004.
4) Barry Wilkinson and Michael Allen. *Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers.* Prentice Hall, 1999.
5) Kento Aida, Wataru Natsume, and Yoshiaki Futakata. Distributed Computing with Hierarchical Master-worker Paradigm for Parallel Branch and Bound Algorithm. In *Proceedings of 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 156–163. IEEE Computer Society, 2003.
6) Makoto Takaki, Keiichi Tamura, Toshihide Sutou, and Hajime Kitakami. New Dynamic Load Balancing for Parallel Modified PrefixSpan. In *ICDE Workshops*, pp. 96–99, 2005.
7) Nicholas Carriero and David Gelernter. How to write parallel programs: a guide to the perplexed. *ACM Computing Surveys*, Vol.21, No.3, pp. 323–357, 1989.
8) Luigi Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, Vol. 27, No. 1, pp. 31–41, 1997.
9) http://kr.expasy.org/prosite/.