Egyetemi doktori (PhD) értekezés tézisei

A Scalable Parallel Algorithm for Decision Support from Multidimensional Sequence Data

Mahdi Esmaeili

Témavezető: Dr. Fazekas Gábor

DEBRECENI EGYETEM
Matematika és Számítástudományok Doktori Iskola

Debrecen, 2011.

Contents

1) Motivation and Aim

Data may be sequential or non sequential in nature. Non sequential data are those data where order of occurrence is not important. Sequential data are those data where order of occurrence is important to consider. Sequential data can be ordered with respect to time or some other dimension such as, space.

A sequential pattern is a subsequence that appears frequently in a sequence database. Mining sequential patterns in large databases has become an important data mining task with broad applications, such as business analysis, web mining, customer shopping sequences, security, and bio-sequences (discovery of motifs in DNA sequences) analysis [1].

In the last decade, a number of algorithms and techniques have been proposed to deal with the problem of sequential pattern mining [2][3][4][5][6]. The main approaches to sequential pattern mining, namely *Apriori*-based and pattern-growth methods are being used as the basis for other structured pattern mining algorithms. However, and despite the fact that pattern-growth algorithms have shown better performance in the majority of the situations, their advantages over *Apriori*-based methods are not sufficiently understood.

It is interesting and useful to mine sequential patterns associated with multidimensional information. In the other words, if sequential pattern mining can be associated with multidimensional information, it will be more effective. While many studies have contributed to find sequential patterns from sequence dataset, there is no significant number of works to mine patterns from multidimensional sequence dataset. This motivates our theses of sequential pattern in multidimensional sequence data.

There could be a large number of sequential patterns in a huge database, especially multidimensional sequence database. Besides, although efficient algorithms have been proposed, mining large datasets requires powerful computational resources. In fact, data mining algorithms working on very large datasets take a very long time on conventional computers to get results. One approach is parallel computing. High performance computers and parallel data mining algorithms can offer a very efficient way to mine very large datasets by analyzing them in parallel.

Parallelize multidimensional sequential pattern mining is the main subject of our thesis and main goal is to introduce data mining techniques on parallel architectures and to show how large scale data mining and knowledge discovery applications can achieve scalability by using systems, tools and performance offered by parallel processing systems. Our approach to efficient sequence mining is parallelization, where the whole computation is broken up into parallel tasks. We explore data parallelism to extract patterns from sequence dataset.

In this study, we have been described a framework for representing multidimensional sequence dataset. Since relational model is one the most popular way to store data, we identify a table as the main dataset with four basic groups of attributes. This model can be simple and comprehensible and originally introduced in [7]. The next step is developing one parallel method that utilizes data parallelism for extracting sequential patterns in multidimensional sequence dataset. From a data mining viewpoint, our method has several main advantages over task parallelism. This simplifies programming and leads to a development time significantly smaller than one associated with task parallel programming, because a lot of previously written serial code can be reused. Our algorithm has a higher degree of machine architecture independence, in comparison with task parallelism. In most applications, the amount of data can increase arbitrarily fast, while the number of lines of code typically increases at a much slower rate.

Although, primary algorithm scales well but we improve this approach with two modification rules. Using a suitable distribution of dataset and an approximate way to find patterns led to have good performance for improved algorithm. In our published papers [8][9], we have been demonstrated that algorithm can greatly reduce the number of scans through the sequence dataset and a good work loading as well. In fact, it tries to optimize local mining at processors while minimizing the data transfer among them.

Our experimental results indicate that primary and improved algorithms have a similar behavior when length of patterns is low. As the length of patterns and database size grow up, improved algorithm scales better than main parallel algorithm but number of patterns generated by improved algorithm is usually much than main method. In fact, improved approach try to use parallel techniques for optimizing the local mining at a worker, and use distributed techniques for construction global patterns or model, while minimizing the communication and data transfers. Consequently, while the primary proposed parallel algorithm scales well, the IMP algorithm is faster than it.

2) Related Works

While many studies have contributed to find sequential patterns from sequence dataset, there has been relatively less works on mining patterns from multidimensional sequence dataset.

According to [10] a multidimensional sequence database is defined over the schema *$(PK, A_1, \ldots, A_n, S)$*, where *PK* is a key, *A_i $(1 \leq i \leq n)$* stands for the dimensions and *S* is in the domain of sequences. A multidimensional sequence takes the form of *$(a_1, \ldots, a_n, s)$*, where *a_i* is in *$\{A_i, *\}$* for *$1 \leq i \leq n$,* and *s* is a sequence. In this paper some methods are proposed that can be classified into two groups: (1) integration of efficient sequential pattern mining and multidimensional analysis methods, and (2) embedding multidimensional information into sequences and using sequential pattern mining method to find patterns. The sequences found by this approach do not contain several dimensions because the time only can be combined with specific dimensions. Consequently, some sequences are not generated.

In [11], two algorithms have been developed. It mines sequential patterns from multidimensional sequence data in the framework of web usage mining. There are some main drawbacks in the work as pointed out by [7]. Three dimensions (page, session, day) consider and these dimensions belong to a single hierarchical dimension. The generated sequences explain correlation between objects over time by considering only one dimension which corresponds to the web pages.

In [7], a definition for multidimensional sequential patterns has been proposed that is close to our model. The authors claim that they aim at considering more than one dimension. This proposition can be extended for approximate values on quantitative dimensions.

3) Basic Definitions

There is a rich variety of sequence terminology but this section defines the concepts sequence, subsequence, sequence pattern, subset, segment, and support; it also discusses major characteristics of our model, multidimensional sequence dataset. Some of the definitions are generic, because different applications have variety properties.

The main table *M* has tuples $<D,Att_1,Att_2,...,Att_n>$, where *D* is an attribute whose its domain is totally ordered and other attributes divided into two groups: analysis and relevant attributes. Notice that irrelevant attributes removed in preprocessing phase. The schema $<D,A_1,...,A_n, R_1,...,R_m>$ is a multidimensional sequence database, where A_i $(1\leq i\leq n)$ are analysis and R_j $(1\leq j\leq m)$ are relevant dimensions. The schema is partitioned into subsets according distinct values of relevant attributes and sequence support computed by number of partitions in which keep corresponding sequence. A subset table *T* is a set of tuples $<D,A_1,A_2,...,A_n>$, where *D* that stated before is an attribute whose its domain is totally ordered, and A_i $(1\leq i\leq n)$ are analysis attributes. A sequence *S* is denoted by an ordered list $<t_1,t_2,...,t_k>$, where t_i is a tuple, i.e., $D(t_1)\leq D(t_2)\leq\ldots\leq D(t_k)$ for $1\leq i\leq k$ and $D(t_i)$=value of *D* tuple t_i. Every tuple has *n* analysis attributes along with an ordered value. A set of analysis attribute values can occur at most once in a same value of *D*, but can occur multiple times in different values of *D* attribute.

The number of distinct values of *D* in a sequence is the length of that sequence. If the length of *S* is *k*, then we call it a *k*-sequence and in similar way *k*-pattern refers to a pattern with length *k*.

A sequence $S_1=<a_1,a_2,...,a_n>$ is called a subsequence of another sequence $S_2=<b_1,b_2,...,b_m>$ and S_2 a super-sequence S_1, if there exist integers $1\leq j_1\leq j_2\leq\ldots\leq j_n\leq m$ such that $a_1=b_{j1}$, $a_2=b_{j2}$,…, and $a_n=b_{jn}$.

Given a minimum support threshold *min_support*, a multidimensional sequence *S* is called a multidimensional pattern if and only if support(*S*)≥*min_support*. The problem of sequential pattern mining is to find the complete set of frequent sequential patterns satisfying a minimum support in the sequence database.

In this framework, $<(t_1, t_2),(t_3),(t_4, t_5, t_1)>$ is a multidimensional *3*-sequence. Brackets and commas may be added to make multidimensional sequences more readable. All tuples in a bracket are assumed to occur at the same time. In first bracket, due to tuples t_1 and t_2 have the same value of D; both of them appear in one bracket ($D(t_1)=D(t_2)$). This way a sequence is an ordered list of brackets in which ordered according to their associated time in a sequence. Number of brackets show length of sequence. Therefore, above sequence is a sequence with length *3* (*3*-sequence). It is obvious that each tuple can be repeated in different brackets (time), such as tuple t_1, which appears in first and third bracket. On the other side, the ordering between tuples in one bracket is not important. According above definition, multidimensional sequence $<(t_2),(t_3)>$ is a subsequence example of sequence $<(t_1, t_2),(t_3),(t_4, t_5, t_1)>$ with length *2*.

A segment is the largest subsequence in which all of its tuples belong to a same time value. As a result, every sequence with length n has n segments. Given a sequence $S=<x_1x_2...x_n>$ with length n, where each x_i ($1 \le i \le n$) consists of one or some tuples with time value i. Each x_i is a segment. For example, in *3*-sequence $S=<(t_1, t_2),(t_3),(t_4, t_5, t_1)>$, there are three segments $<(t_1, t_2)>$, $<(t_4, t_5, t_1)>$, and $<(t_3)>$,.

Let us consider a real example of taking the course by students. Suppose we have a relation *Marks(id, course, score, year/semester)* in which we want to record for each student their identify number, course name, score, and year/semester. Each tuple (d,c,s,y) indicate that student with identify number d has taken course c in semester y and obtained score s. As you can suggested, we set *id* as relevant attribute, *year/semester* as ordered attribute, and other columns *(course* and *score)* as analysis attributes. According to *id* column values, main table *Marks* partition into subsets that each one holds information about specific student. Finding the patterns may help to capture some ordering of students' course taking. Following patterns are several useful and important pattern results.

- More than 60 percent of students fail Programming course.
- While Operating System is not prerequisite of Database but at least 80 percentages of students take it before Database.
- Students who take Data Mining always observe following ordering: Programming → Data Structure → Database → Data Mining

4) Primary Proposed Parallel Algorithm

We will first describe a standard algorithm (STA) for mining multidimensional sequence patterns within the entire sequence data. We use data parallelism such that subsets are distributed among the workers and each worker performs same operation on it.

Let us first analyze search space before discussing about main method. Given a subset with length c (c-sequence), and suppose that each element of vector d_i for $1 \leq i \leq c$ shows number of tuples with equal time dimension. Number of sequences with length *1,2,… ,c* for this subset computed as follows:

Number of sequences with length 1: $\sum_{i=1}^{c}(2^{d_i}-1)$

Number of sequences with length 2: $\sum_{i=1}^{c-1}(2^{d_i}-1)\sum_{j=i+1}^{c}(2^{d_j}-1)$

Number of sequences with length 3: $\sum_{i=1}^{c-2}(2^{d_i}-1)\sum_{j=i+1}^{c-1}(2^{d_j}-1)\sum_{k=j+1}^{c}(2^{d_k}-1)$

Number of sequences with length c: $\prod_{i=1}^{c}(2^{d_i}-1)$

As you observe, search space is quite large and the serial algorithms are not scalable. In parallel environment, each worker scans datasets to find patterns with length one (*1*-patterns) according to the predefined support threshold. Next steps generate the sequences by joining the patterns in the previous step. Dataset is scanned to check the support of candidates. We can use *Apriori* property to develop breadth first search algorithm to find sequence patterns. The main idea is that, if a sequence is not pattern, we do not looking for any super-sequence of it. Schematic diagram of primary proposed method shows in Figure 1. It is obvious that phases 1 and 2 have been repeated once and phase 3 repeats until there is no sequence to be mined.

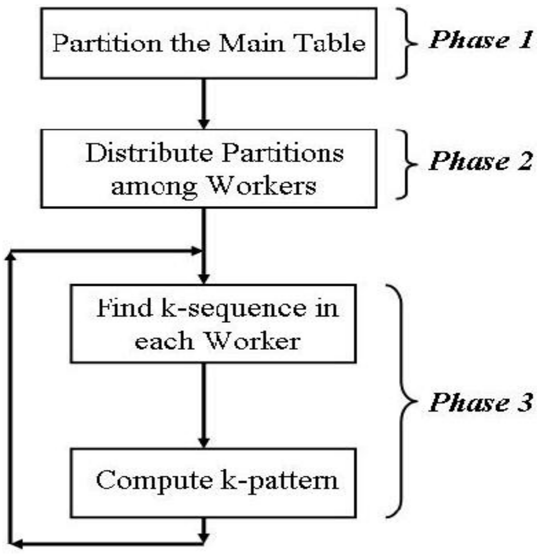

Partition the Main Table
Phase 1
Distribute Partitions among Workers
Phase 2
Find k-sequence in each Worker
Compute k-pattern
Phase 3

Figure 1: Schematic diagram of proposed algorithm

Our algorithm conducts a level-by-level candidate generation and test pruning following the *Apriori* property in workers independently. At each level, only potentially frequent candidates are generated and tested.

4.1) Partition the Main Table

In first phase of algorithm, one worker (processor) known as coordinator, partitions table *T* into *P* sequence datasets (subsets). For our proposed algorithm, this process does in the three following steps:

I. The main table should be partitioned into subsets according to current values of relevant attributes.
II. We assign a simple code to each combination of current analysis attribute values. It is easier and faster for algorithm to works with smaller dataset.
III. Merge tuples with equal time value into identical segment.

4.2) Subsets Distribution

In this phase, coordinator distributes *m* subsets among *n* workers regardless of length of subsets. This way, $subset_i$ is assigned to worker $P_{i \bmod n}$, for $i=0,\dots,m-1$. Thus, worker P_j owns the subsets $j, j+n, \dots, j+n.(\lceil m/n \rceil - 1)$ for $j<m \bmod n$ and $j, j+n, \dots, j+n.(\lceil m/n \rceil - 2)$ for $(m \bmod n) \leq j < n$.

For example when we set $n=4$ and $m=10$ the subsets distribution results are such that P_j for $j<2=(10 \bmod 4)$ owns the subsets $j, j+4, j+4*(3-1)$ and P_j for $2 \leq j < 4$ owns the subset $j, j+4*(3-2)$.

Following we can see in this example what subset goes to which processor:

P_0: owns *$subset_0$, $subset_4$, $subset_8$*

P_1: owns *$subset_1$, $subset_5$, $subset_9$*

P_2: owns *$subset_2$, $subset_6$*

P_3: owns *$subset_3$, $subset_7$*

4.3) Patterns Mining

In iteration *k*, every worker mines all *k*-sequence and computes its support (local support) and then sends to coordinator. Coordinator collects results and discards some sequences in which its support is less than *min_support* and sends *k*-patterns to workers.

While the general procedure of this phase is simple, the new sequence generation is non trivial. As mentioned previously, every worker scans subsets once to find all *1*-sequence and computes local support and then sends to coordinator. Coordinator collects results and sends back *1*-patterns to workers. After generating *1*-patterns, the set of candidate *k*-sequences (for $k>1$) are generated by joining the *(k-1)*-patterns found in the previous step. New scans of each subset collect the support for candidates and then send to coordinator to find the new set of patterns. Two patterns with length *k* can be joined if (i) patterns have equal length and (ii) value of time dimension *k-1* segments of two patterns are identical.

We assume that $S_x=<x_1x_2...x_{k-1}x_k>$ and $S_y=<y_1y_2...y_{k-1}y_{k+1}>$ are two *k*-patterns ($k>1$), where each x_i ($1\leq i\leq k$) and y_j ($1\leq j\leq k+1, j\#k$) are a tuple or consists of some tuples with identical value of time. Without loss of generality, we assume that subscripts show time dimension value for segments. As you know, both patterns have equal length (*k*) and based on subscripts of each segment, values of time *k-1* segments of two patterns are same. The conditions satisfied and join of S_x and S_y a new *(k+1)*-sequence is generated: $S_{xy}=<(x_1y_1)(x_2y_2)\ldots(x_{k-1}y_{k-1})(x_k)(y_{k+1})>$. While in each segment the ordering of tuples is not important ($x_iy_i=y_ix_i$), duplication of tuples is not allowed. In other words, when segments x_i and y_i are merged, every tuple appears once and repetitions are removed.

5) IMP: Improved STA Algorithm

The STA algorithm can be improved further. The number of candidate patterns of a certain level is typically an exponential function of number of discovered lower level. Consequently, coordinator became a bottleneck. Message passing between workers and coordinator can be heavy especially when number of patterns and their length grow up. Besides, we usually have no good load balance and the more workers are idle. Since the time cost of the passes over the datasets plays an important role to performance of algorithm, we made two following rules to improve STA algorithm.

5.1) Rule 1: Distribution

Using a suitable data distribution can be useful to obtain good load balancing. This simplifies the parallel programming and supports a good performance. The load imbalance is generally defined as:

$Load\ Imbalance=Max\ (RT_1,\ldots,RT_p)/Min(RT_1,\ldots,RT_p)$, Where RT_i is runtime on worker or processor *i*. Ideally, this value is very nearly to 1. The larger the load imbalance, the more processors are idle. The main goal of this improvement rule is to assign the subsets such that a good load balancing results.

In improved algorithm (IMP), we present a cyclic data distribution. Cyclic distributions are one of the most useful distributions for increasing the performance of embarrassingly parallel computations. The set of processors is denoted as $P=\{P_0,P_1,\ldots,P_{n-1}\}$ and the subsets is denoted as $S=\{S_0,S_1,\ldots,S_{m-1}\}$, where S_i is a subset, i.e., Length(S_0)≤Length(S_1)≤… ≤Length(S_{m-1}) for $0\leq i\leq m-1$. We assign subsets to processors in round robin two ways so that subset S_i is assigned to processor $P_{(i \bmod 2n)}$ for $(i \bmod 2n)<n$, and processor $P_{(2n-1)-(i \bmod 2n)}$ when $(i \bmod 2n)\geq n$. Figure 2 shows how to distribute m subsets among n processors.

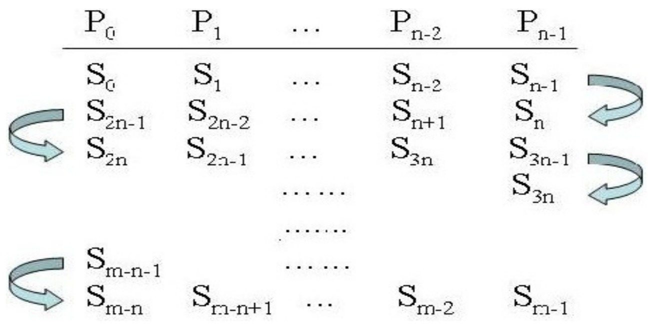


Figure 2: Distribution of *m* subsets among *n* processors

For the example $n=4$ and $m=18$ the subsets distribution

P_0: owns $subset_0, subset_7, subset_8, subset_{15}, subset_{16}$

P_1: owns $subset_1, subset_6, subset_9, subset_{14}, subset_{17}$

P_2: owns $subset_2, subset_5, subset_{10}, subset_{13}$

P_3: owns $subset_3, subset_4, subset_{11}, subset_{12}$

Results, where $subset_j$ for $(j \bmod 8)<4$ belong to processor $P_{(j \bmod 8)}$, and send to processor $P_{7-(j \bmod 8)}$ when $(j \bmod 8)\geq 4$.

This distribution is only based on length of subsets. We know that each subset with length k usually has cardinality (number of tuples) more than k. If we can use length of subset along with cardinality to distribute subsets, we will achieve more performance, surly.

5.2) Rule 2: Efficient Pattern Mining

In algorithm STA, the check process is quite straightforward; by scanning the data the support counts of these candidate sequences can be obtained. By comparing them with the support threshold we can get those frequent sequential patterns.

In IMP approach, our strategy first constructs the *2*-sequences and *3*-sequences and then mine whole of *2*-patterns and *3*-patterns, similar to STA algorithm. After this, each worker generates sequence with length larger than three and discover patterns, independently. Every worker uses all of patterns with length two and three to generate new patterns. The algorithm will iteratively generates new candidate *(k+1)*-sequence using the frequent sequence with length *k* (*k*-pattern) found in the previous iteration.

Given sequence $S_x=<x_1x_2...x_{k-1}x_{k1}>$ and $S_y=<y_1y_2...y_{k-1}y_{k2}>$ are two *k*-patterns ($k>1$), where each x_i and y_i (*i is in {1,2,…,k-1,k1,k2}*) are a tuple or consists of some tuples with identical value of time, and subscripts show time value. Thus, there are *(k-1)* segments in which have equal time value. New sequence *S* generates depends on ordering between *k1* and *k2*.

$$S=\begin{cases} S_{xy}=<(x_1y_1)(x_2y_2)\ldots(x_{k-1}y_{k-1})(x_{k1})(y_{k2})> & \text{if } k1<k2 \\ S_{yx}=<(x_1y_1)(x_2y_2)\ldots(x_{k-1}y_{k-1})(y_{k2})(x_{k1})> & \text{if } k1>k2 \end{cases}$$

According to second improvement rule, the new sequence S_{xy} (or S_{yx}) is a pattern with length *(k-1)* if two following conditions are satisfied.

(1) The *2*-sequence $<(x_{k1})(y_{k2})>$ (or $<(y_{k2})(x_{k1})>$) is pattern.

(2) The 3-sequences $<(x_iy_i)(x_{k1})(y_{k2})>$ (or $(x_iy_i)(y_{k2})(x_{k1})$), where $1 \leq i \leq k-1$ are pattern.

For example, consider the case where $S_x=<(ab)_1,(a)_4,(cde)_5>$ and $S_y=<(b)_1,(d)_3,(bcd)_5>$. New sequence generates $S=<(ab)_1,(d)_3,(a)_4,(cdeb)_5>$. While STA algorithm scan all of subsets, and coordinator collects their support value to check S is pattern or not, in IMP algorithm according to second rule, we must test 2-sequence $<(d)_3,(a)_4>$ and two 3-sequences $<(ab)_1,(d)_3,(a)_4>$ and $<(d)_3,(a)_4,(cdeb)_5>$ are patterns or not. If so, S is pattern otherwise S does not satisfy the rules and is not a pattern.

In modified algorithm IMP, each worker keeps two tables consist of 2-patterns and 3-patterns. This way, algorithm must check at most $k-1$ sequences to test one k-sequence may be pattern or not. Consequently, every worker is able to check whether new sequence is pattern or not, independently. In other words, after mining patterns with length two and three, communication between coordinator and workers will be terminated.

Theoretically, the all of identified patterns using this rule are not guaranteed to be valid because we only consider some subsequence of candidate and do not count support of corresponding patterns. Number of extra patterns is not high and in near all our experiments, number of patterns is equal. In other words, there are no any extra patterns when we have sequences with length less than 15. In fact, using this rule is made to generate further patterns but we will see that number of extra generated patterns increase slowly when length of patterns grows. Experiments in the next section demonstrate the power of these two modification rules.

6) Experimental Results

All algorithms were implemented in MATLAB with Parallel Computing Toolbox. Our machine environment consists of a network of 8 computers (2.6 GHz CPU) with 2 GB memory running Windows XP operating system.

6.1) Sequence Datasets

To gain insight on how algorithms behave under various setting, we choose two most common used resources UCI Repository and IBM Data Generator.

The UCI Machine Learning Repository is a collection of real databases that are widely used by the researchers for the empirical analysis of algorithms [12]. For evaluation of our algorithms in practical experiment, we selected four datasets from UCI Repository. The training datasets were chosen to include a wide range of domains and are summarized in Table 1. Datasets are used with some modifications because we must adjust the datasets to suit our multidimensional sequence model. For example, the chosen dataset may be contained of attributes with missing values or as mentioned previously we need at least one dimension whose its domain is totally ordered.

Table 1: Real dataset descriptions for test

Name of Dataset	Data Types	Number of Instances	Number of Attributes
Adult	Multivariate	48842	14
Localization Data for Person Activity	Univariate, Sequential, Time series	184860	8
Spoken Arabic Digit	Multivariate, Time series	8800	13
Thyroid Disease	Multivariate, Domain Theory	7200	21

We also used IBM Synthetic Data Generator with minor modification. Table 2 summarizes the main parameters of the datasets. For the purpose of evaluation of the algorithms, we use some synthetically generated sequences, based on parameters shown in Table 2.

Table 2: The Parameters of the Datasets Generator

Notation	Meaning	Default Setting
N_{seq}	Number of subsets (sequences)	100,000
N_{tra}	Average number of time values per subset	10
L_{tra}	Average number of tuples with identical time value	2.5
N_{tup}	Number of different tuples	10,000
R	Repetition level for tuples	0

6.2) Discussion

We ran many experiments with various datasets. We obtain results over several criteria including minimum support, size of dataset, number of workers, number of sequence (pattern), length of sequence (pattern), and maximum length of sequence (pattern).

Figure 3 plots the effect of minimum support over number of patterns to be mined. The first observation is that, as the minimum support was increased, number of patterns reduced quickly. This shows why it is hard to find patterns when minimum support is low. It is also clear that maximum length of patterns grow up when value of minimum support reduced (Figure 4).

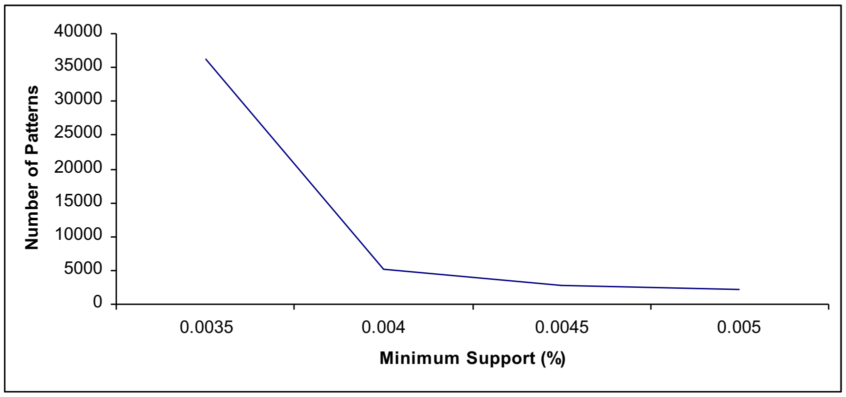


Figure 3: Effect of minimum support value over number of patterns

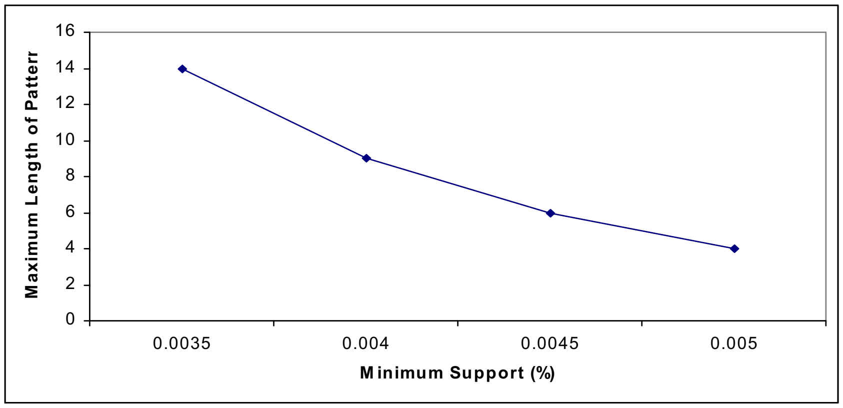


Figure 4: Effect of minimum support threshold over length of patterns

We have also studied the effect of changing minimum support on the STA algorithm performance. We used 1, 2, 4, and 8 workers, and support threshold ranges from 0.0035% to 0.0050%. Figure 5 shows execution time over minimum support in 4 statuses. It can be seen that STA algorithm scales well but algorithm has to deal with longer patterns and data transfer between coordinator and workers extremely grow up. The coordinator becomes a bottleneck and major cost is counting support value for sequences in which send to coordinator. Even in this case (Figure 5), STA speed up is obvious and scales almost linearly. But it seems using more than 4 processors is not as well as less than that.

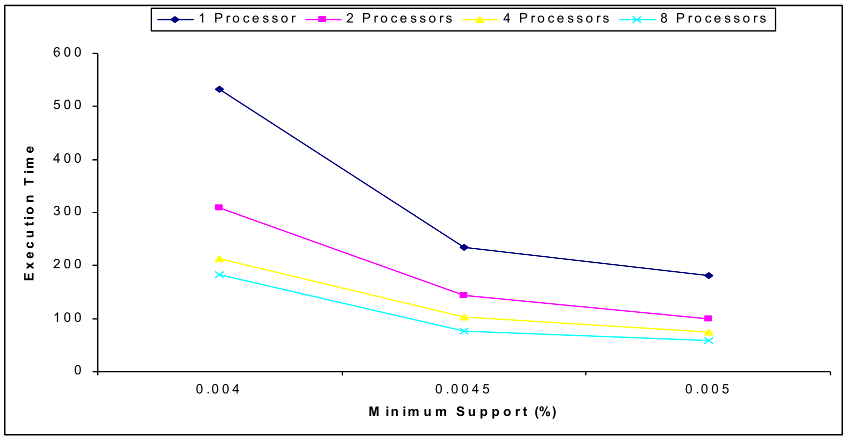


Figure 5: Effect of minimum support over execution time

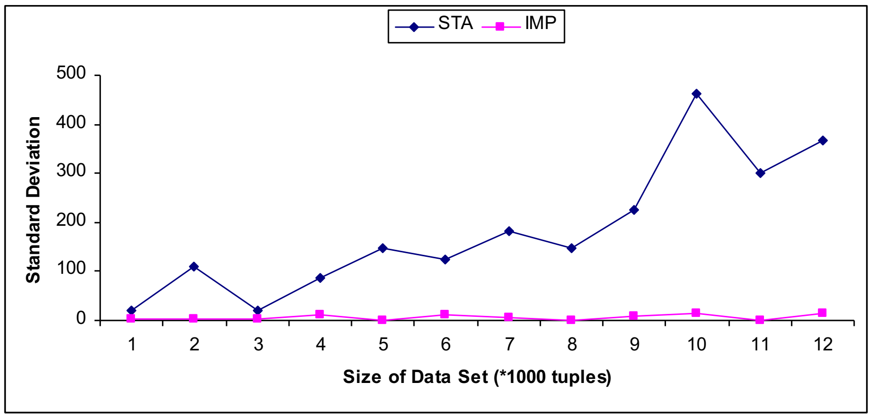
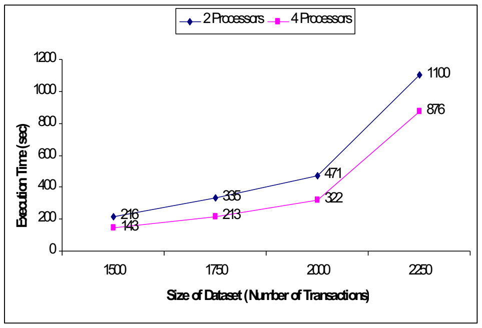
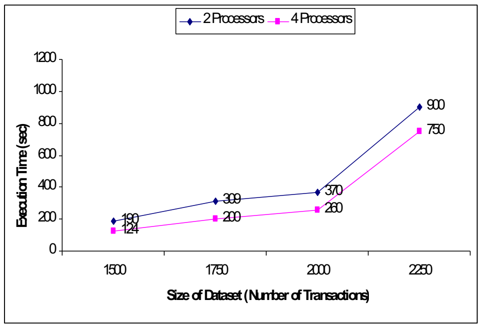


Figure 6: Comparison of subsets distribution

Figure 6 compares subsets distribution of STA and IMP approaches when the size of dataset (number of transaction) changes. The number of workers is set to 4. After distribution of subsets among 4 workers, we add length of subsets or sequences in which assign to each worker and then calculate standard deviation. As the dataset becomes longer, standard deviation of STA algorithm grows up while there is no considerable variation in standard deviation of IMP algorithm. We observe that the work balancing of IMP in terms of length of sequences is better than STA. The above analysis indicates that IMP is effective and scalable in mining large database.





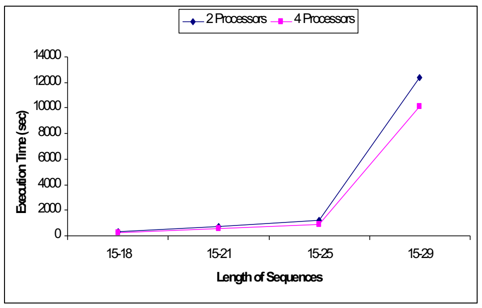
(a) (b)

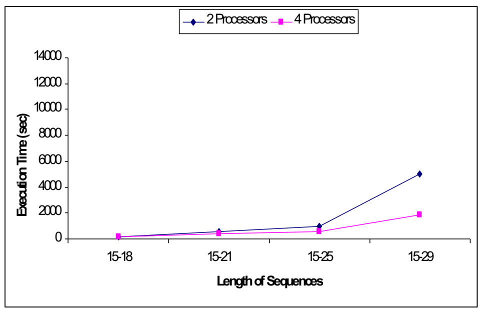
Figure 7: Execution time over database size

Figure 7 plots execution time over dataset size for two approaches separately. The database size changes from 1500 to 2250 and the support threshold set to 0.004%.

Figure 7a shows the performance of STA algorithm when it runs with 2 and 4 processors. Figure 7b shows the scalability of IMP algorithm since execution time grows almost linearly when the size of database (number of transactions) increases. We observe that the performance of IMP in terms of scalability and runtime is better than STA.

One of the most important criteria to consider two approaches is length of sequences. We test the effect of this parameter over execution time. The results are shown in Figure 8a and 8b for STA and IMP algorithms, respectively. The length of sequences rang from 15 to 29, database size set to 2000 transactions and minimum support value is 0.004%.



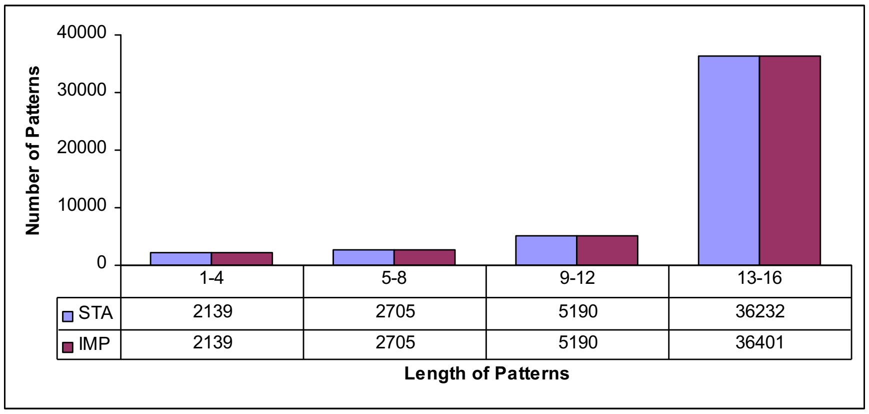


(a) (b)

Figure 8: Effect of length of sequences on execution time

As expected IMP algorithm is faster when length of sequences grows. This is because STA has to deal with longer sequences and patterns when dataset contains of longer patterns. While the algorithm IMP rather than scan the datasets many times for longer patterns, it only use a dataset with all of *2*-patterns (patterns with length 2) and *3*-patterns in which every worker keep it.

As stated before, theoretically the number of generated patterns with algorithm IMP is usually much than that with algorithm STA. In near all our experiments, number of patterns is equal. Figure 9 shows number of patterns when length of patterns changes. We use real datasets and generate some synthetic datasets with size 2000, and minimum support changes from 0.0035 to 0.005.



	1-4	5-8	9-12	13-16
▪ STA	2139	2705	5190	36232
▪ IMP	2139	2705	5190	36401

Figure 9: Number of generated patterns by two approaches

As the length of patterns become larger, there could be many generated patterns by IMP algorithm which are not patterns but it grows slowly. When length of patterns is between 1 and 12, number of patterns generated by two algorithms is same. Number of patterns goes from 36232 to 36401 when length of patterns is between 13 and 16. The results in Figure 9 show that difference of patterns number for sequences with length 16 is only 169 patterns. While in real multidimensional dataset we rarely found patterns with length more than 16.

7) Summary

Multidimensional mining has been attracting attention in recent research into data mining. Very large search space and data volume have made many problems for serial algorithms to mine sequential patterns. In order to effectively mine, efficient parallel algorithm is necessary. In addition, parallelism often can not be fully used because of communication and processing limits.

In the main part of this dissertation, we first theoretically utilize a multidimensional sequence model and then use SPMD (Single Program Multiple Data) strategy to parallelize multidimensional sequential pattern mining. According to this method a set of processors execute in parallel the same algorithm on different partitions of a dataset. From a data mining viewpoint, our approach has several main advantages over task parallelism. This simplifies programming and leads to a development time significantly smaller than one associated with task parallel programming, because a lot of previously written serial code can be reused. It has also a higher degree of machine architecture independence, in comparison with task parallelism. In most applications, the amount of data can increase arbitrarily fast, while the number of lines of code typically increases at a much slower rate. To put it in simple terms, the more data is available, the more opportunity to exploit data parallelism.

The main goal of the algorithm is balanced workload among the processors and good scalability. We use two modification rules to improve the main parallel algorithm. Using a suitable distribution of dataset and an approximate way to find patterns led to have good performance for improved algorithm. In other words, an attractive property of the improved algorithm is good behavior and scalability when length of patterns grows up while main parallel algorithm faced to very high computation time in this situation.

We have implemented our parallel algorithm using MATLAB Parallel Computing Toolbox and several datasets on a network with 8 workstations. In summary our approach can greatly reduce the number of scans through the sequence dataset by only examining a small data of patterns with length two and three and a good work loading as well. In fact, it tries to use parallel techniques for optimizing the local mining at a worker, and uses distributed techniques for construction global patterns or model, while minimizing the amount of results communicated.

Our experimental results indicate that primary and improved algorithms have a similar behavior when length of patterns is low. As the length of patterns and database size grow up, improved algorithm scales better and faster than primary parallel algorithm but number of patterns generated by improved approach is usually much than main parallel algorithm. Consequently, while the primary proposed parallel algorithm scales well, the IMP algorithm is faster than it.

8) Future Works

It is easy to see that the first scan through the dataset to *1*-sequence construction takes so much computation. Using some techniques such as sampling can be efficient for mining *1*-sequence and reduce time for next steps. It is obvious that using a suitable data structure can improve our approach. Besides, if we can use length of subset along with cardinality to distribute subsets, we will achieve more performance, surly.

The future research issues in multidimensional sequential pattern mining can be: multidimensional sequential pattern mining with constraints, interactive multidimensional sequential pattern mining or multidimensional sequential pattern mining integrated with taxonomies and hierarchies.

References

[1] J. Han and M. Kamber. *Data Mining Concepts and Techniques.* Morgan Kaufmann, Edition 2, 2006.

[2] R. Agrawal and R. Srikant. *Mining Sequential Patterns.* International Conference on Data Engineering (ICDE'95), pp 3–14, Taipei, Taiwan, March 1995.

[3] R. Srikant and R. Agrawal. *Mining Sequential Patterns: Generalizations and Performance improvements.* 5th International Conference on Extending Database Technology (EDBT'96), pp 3–17, Avignon, France, March 1996.

[4] M. J. Zaki and C. J. Hsiao. *CHARM: An Efficient Algorithm for Closed Itemset Mining.* SIAM International Conference on Data Mining (SDM'02), pp 457–473, Arlington, VA, April 2002.

[5] M. J. Zaki. *SPADE: An Efficient Algorithm for Mining Frequent Sequences.* Machine Learning, Vol. 40, pp 31–60, 2001.

[6] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu. *Mining Sequential Patterns by Pattern-growth: The Prefixspan Approach.* IEEE Transaction on Knowledge and Data Engineering, Vol. 16, pp 1424–1440, 2004.

[7] M. Plantevit, Y. W. Choong, A. Laurent, D. Laurent, and M. Teisseire. *M2SP: Mining Sequential Patterns Among Several Dimensions.* Principles of Knowledge Discovery in Databases, Vol. 3721, pp 205–216, 2005.

[8] M. Esmaeili and M. Tarafdar. *Application and Limitation of Parallel Modeling in Multidimensional Sequential Pattern.* International Conference

on Electrical, Computer, Electronics and Communication Engineering, pp 199–202, Paris, France, October 2010.

[9] M. Esmaeili and M. Tarafdar. *Sequential Pattern Mining from Multidimensional Sequence Data in Parallel.* International Journal of Computer Theory and Engineering (IJCTE), Vol. 2, Issue 5, pp 733–738, October 2010.

[10] H. Pinto, J. Han, J. J. Pei, K. Wang, Q. Chen, and U. Dayal. *Multidimensional Sequential Pattern Mining.* Conference on Information and Knowledge Management, pp 81–88, 2001.

[11] C. C. Yu and Y. L. Chen. *Mining Sequential Patterns from Multidimensional Sequence Data.* IEEE Transactions on Knowledge and Data Engineering archive, Vol. 17, Issue 1, pp 136–140, 2005.

[12] C. L. Blake and C. J. Merz. *UCI Repository of Machine Learning Databases.* Irvine, CA: University of California, Department of Information and Computer Science, 2011. [http://archive.ics.uci.edu/ml/datasets.html]

Publications

Referred Journal and Conference Proceeding Papers

[1] M. Esmaeili and G. Fazekas, "*Parallel Mining of Sequential Patterns from Multidimensional Sequence Data*", International Journal of Computer Science Issues(IJCSI), 2011, Accepted for publication.

[2] M. Esmaeili and A.H. Mousavi, "*Variable Reduction for Multi-objective Optimization Using Data Mining Techniques: Application to Aerospace Structures*", 2nd International Conference on Computer Engineering and Technology, China, pp 333–337, ISBN 978–1–4244–6348–0, April 2010.
Digital Object Identifier: 10.1100/ICCET.2010.5486051, Index by: IEEE Xplore. INSPEC Accession Number: 11521830.

[3] M. Esmaeili and M. Tarafdar, "*Application and Limitation of Parallel Modeling in Multidimensional Sequential Pattern*", International Conference on Electrical, Computer, Electronics and Communication Engineering, pp 199–202, ISSN 1307–6892, Paris, France, October 2010.
Word Academy of Science, Engineering and Technology:702010 Indexed by: Scopus, Engineering Index (Compendex), and CiteSeerX

[4] M. Esmaeili and M. Tarafdar, "*Sequential Pattern Mining from Multidimensional Sequence Data in Parallel*", International Journal of Computer Theory and Engineering (IJCTE), Vol. 2, Issue 5, pp 733–738, ISSN 1793–8201, October 2010.
http://www.ijcte.org/papers/233-G354.pdf Indexed by Google Scholar

[5] M. Esmaeili and G. Fazekas, "*Finding Sequential Pattern from Large Sequence Data*", International Journal of Computer Science Issues (IJCSI), Vol. 7, Issue 1, ISSN 1694–0814, pp 43–46, January 2010.
Cite as: arXiv:1002.1150v1 [cs.DB] Indexed by: CiteSeer, INSPEC and DOAJ

[6] M. Esmaeili and G. Fazekas, "*Feature Selection as an Improving Step for Decision Tree Construction*", International Conference on Machine Learning and Computing (ICMLC'09), pp 35–39, Australia, July 2009.
Indexed by: INSPEC and ISI Web of Knowledge

International Conference

[7] M. Esmaeili and G. Fazekas, "*Using Decision Tree as a Feature Selector for other Methods*", 7th Annual International Conference on Computer Science and Information Systems, Athens, Greece, June 2011.

[8] M. Esmaeili and M. Tarafdar, "*Attribute Importance Ranking for Classification Algorithms*", 4th International Data Mining Conference (IDMC), Sharif University of Technology, Tehran, Iran, pp 4–9, December 2010.

[9] M. Esmaeili and M. Tarafdar, "*Efficient Techniques of Data Selection for Classification*", 3rd International Data Mining Conference (IDMC), Iran University of Science & Technology, Tehran, Iran, pp 120–124, November 2009.

National Conference and Magazine

[10] M. Esmaeili, M. Babavalian, N. Esfandiary, and M. Tarafdar, "*Multi–Objective Genetic Algorithm for Association Rule Mining*", 8th Iranian Conference on Fuzzy Systems & 9th Conference on Intelligent Systems, University of Malek-e-Ashtar, Iran, pp 58–67, October 2008.

[11] M. Esmaeili, "*Survey on Data Mining Techniques*", Web Magazine, Tehran Institute of Technology, Vol. 79, pp 46–49, January 2007.

[12] M. Esmaeili, "*Data Preprocessing for Data Mining*", Web Magazine, Tehran Institute of Technology, Vol. 79, pp 18–23, December 2006.

Research Reports

[13] "*Analysis and Modeling of Internet User Behavior Using Data Mining Techniques*", 2010/2011.

[14] "*Data Discretization with Optimal Binning and Ordinal Regression*", 2008/2009.

[15] "*Using Genetic Algorithm for Rule Generation (Classification)*", 2008/2009.

[16] "*An Efficient Method for Feature Subset Selection*", 2007.

[17] "*Survey on Data Preparation for Data Mining Techniques*", 2006.

Books

[18] "*File System and Structures*", 2008. (Published in Persian Language Titled " سیستم و ساختار فایلها")

[19] "*Q&A of Data Base*", 2007. (Published in Persian Language Titled" پرسش و پاسخ های پایگاه داده ها")

Talks

[20] "*Using Decision Tree as a Feature Selector for other Methods*", 7th Annual International Conference on Computer Science and Information Systems, Athens, Greece, June 2011.

[21] "*Application and Limitation of Parallel Modeling in Multidimensional Sequential Pattern*", International Conference on Electrical, Computer, Electronics and Communication Engineering, Paris, France, October 2010.

[22] "*Application of Computer Science in Medicine*", Kashan University of Medical Science and Health Services, August 2007.

[23] "*International Computer Driving License (ICDL) Certifications*", Governor of Kashan, November 2006.

[24] "*Computer Science and Education*", Ministry of Education, July 2005.