Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Engineering

Elixir Comp. Engg. 81 (2015) 31915-31919

ISSN: 2229-712X

A Genetic Algorithm Based Approach to Closed Sequential Pattern Mining

V. Purushothama Raju[1] and G.P. Saradhi Varma[2]

[1]Research Scholar, Department of CSE, Acharya Nagarjuna University, Guntur, India.

[2]Department of IT, S.R.K.R. Engineering College, Bhimavaram, India.

ARTICLE INFO

Article history:

Received: 12 March 2015;

Received in revised form:

15 April 2014;

Accepted: 21 April 2015;

Keywords

Data mining,

Sequential pattern mining,

Closed sequential pattern mining,

Genetic algorithm

Sequence database.

ABSTRACT

Closed sequential pattern mining has attracted increasing concerns in recent data mining research because it is more efficient than sequential pattern mining and produces more compact result set. Genetic algorithms perform global search and have less time complexity compared to other algorithms. In this paper, we propose a novel algorithm GCSP for mining closed sequential patterns using genetic approach. It uses an efficient fitness function to improve the performance. The results show that the proposed algorithm GCSP can find closed sequential patterns efficiently and outperforms CloSpan and ClaSP.

Introduction

Data mining has attracted the information industry and the society in recent years, due to the availability of large amounts of data and the requirement for converting such data into useful information and knowledge. The knowledge obtained can be utilized in different applications such as market analysis, customer retention, finance, insurance, production control and fraud detection.

The concept of sequential pattern mining was first introduced by R. Agrawal and R. Srikanth in [1], and aimed at finding sequential patterns in a sequence database, given a user-specified minimum support threshold. There are several applications of sequential pattern mining including mining customer shopping sequences, DNA sequences and Web click streams. Closed sequential pattern mining was introduced to eliminate the drawbacks of sequential pattern mining algorithms. Closed sequential pattern mining produces more compact result set than sequential pattern mining and also offers better efficiency for mining long sequences.

In general closed sequential patterns are produced from large data sets by applying algorithms like CloSpan, ClaSP and etc, which take more execution time to find all the closed sequential patterns. By using Genetic Algorithm (GA) we can reduce the execution time. The major advantage of using GA in the discovery of closed sequential patterns is that it performs global search and its time complexity is less compared to other algorithms as the genetic algorithm is based on the greedy approach.

Genetic algorithms are adaptive heuristic search algorithms based on the evolutionary approaches of natural selection and genetic. The basic idea of GAs is to simulate methods in natural system necessary for evolution, particularly those that follow the rules first laid down by Charles Darwin's survival of the fittest.

As such they correspond to an intelligent utilization of a random search within a distinct search space to resolve a problem.

GAs provide alternative methods for solving the problem and outperform other traditional methods in most of the problems. Genetic algorithms are easy to develop and validate which makes them attractive for a large number of applications. GAs have been effectively applied in many search, optimization and machine learning problems. Most of the real world problems concerned determining optimal parameters, which might prove hard for traditional methods but perfect for GAs.

Basically, genetic algorithms are solutions to optimization or search problems by means of simulated evolution. Methods based on natural selection, crossover, and mutation are frequently applied to a population of binary strings which correspond to potential solutions. Over time, the number of above average individuals increases and highly fit individuals are combined with several fit individuals to find good solutions to the problem at hand.

Use of mutation makes the method suitable for finding global optimal solution, even in difficult problem domains. GAs do not need information about the distribution of the data and can efficiently explore the search space of possible solutions. We make use of the ideas of GA to focus on the search space with good solutions because this always finds more frequent patterns first. GA is more useful than methods which treat all candidates equally, particularly at low support thresholds.

In this paper, we propose a novel algorithm GCSP for mining closed sequential patterns from a given sequence database using genetic approach. It uses an efficient fitness function to improve the performance. The results show that the proposed algorithm GCSP can find closed sequential patterns efficiently and has remarkable performance compared with CloSpan and ClaSP.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 briefly introduces the basic concepts of genetic algorithms. Section 4 presents the proposed method. Section 5 reports the performance evaluation. Finally, we conclude the work in Section 6.

Related Work

Agrawal and Srikant [1] introduced the problem of sequential pattern mining. Later, several algorithms were proposed for sequential pattern mining, the efficient algorithms

Tele:

E-mail address: prajusvs@gmail.com

are SPADE[2], PrefixSpan[3] and SPAM[4]. SPADE adopts breadth-first search where as PrefixSpan and SPAM adopt depth-first search. SPADE adopts a vertical data format and mines the sequential patterns through a simple join on id-lists. PrefixSpan adopts a horizontal data format and mines the sequential patterns under the pattern growth paradigm. SPAM mines sequential patterns using vertical bitmap representation and it outperforms PrefixSpan and SPADE on large datasets. However, SPAM requires more memory than the other two methods.

In recent years, some research has started to focus on closed sequential pattern mining. There are only a few popular algorithms CloSpan [5], BIDE[6] and ClaSP[7] in closed sequential pattern mining. CloSpan produces a candidate set for closed sequential patterns and performs post pruning on it. CloSpan requires more storage to store the closed sequence candidates when mining long patterns or the support threshold is low and it offers poor scalability. BIDE adopts the framework of PrefixSpan and uses BackScan pruning method to stop growing redundant patterns. BIDE is a computational intensive approach since it requires more number of database scans for the bi-direction closure checking and the BackScan pruning.

ClaSP mines closed sequential patterns in temporal transaction data. It is inspired on the Spade algorithm that uses vertical database format strategy for generating sequential patterns. ClaSP has two main phases: The first phase produces Frequent Closed Candidates (FCC) that is kept in main memory; and the second phase performs a post-pruning to remove all non-closed sequences from FCC to finally get exactly frequent closed sequences.

A. Bilal and A. Erhan [8] proposed a genetic algorithm based method to mine negative quantitative association rules. They used an adaptive mutation probability and an adjusted fitness function. Sandra de Amo et al.[9] used a genetic algorithm to mine generalized sequential patterns but it is based on SQL expressions. S.Ghosh et al. [10] proposed a genetic algorithm for mining frequent itemsets. They defined a performance measure based on Apriori algorithm. B. Bidgoli et al.[11] presented a genetic algorithm for classifying students in order to predict their final grade based on features extracted from logged data in an education web-based system.

Basic Concepts of Genetic Algorithms

Standard GA uses genetic operators such as selection, crossover and mutation to compute a new generation of strings. GA creates improved solutions for successive generations. The probability of selecting an individual is proportional to the goodness of the solution it represents. Hence the quality of the solutions in successive generations is improved. The process is terminated when an acceptable or optimum solution is found. Selection operator deals with the probabilistic survival of the fittest. Only the chromosomes with high fitness value are selected to survive. The fitness of a chromosome α is measured according to its support.

The crossover operator chooses genes from parent chromosomes and produces a new offspring. The crossover operation is performed over two chromosomes as follows. Random positions P1, P2 in the first and second chromosomes are selected for crossover. The portion of the first chromosome on the left side of P1 is exchanged with the portion of the second chromosome on the right side of P2 to create child1 and the portion of the first chromosome on the right side of P1 is exchanged with the portion of the second chromosome on the left side of P2 to create child2.

The mutation operator is used to avoid falling all solutions of a population into a local optimum. Mutation alters randomly the new offspring. The mutation is performed over a chromosome α in the following way. An arbitrary element in α is chosen and it is replaced with a different element of the same kind. For example, the bits in the chromosome are changed randomly from 1 to 0 or from 0 to 1.

Genetic algorithm is a general purpose search algorithm which uses principles motivated by natural genetic populations to grow solutions to problems. All GAs typically begins from a set, known as population, of random solutions. These solutions are developed by the repeated selection and variations of more fit solutions, following the rule of survival of the fittest. The components of the population are called individuals or chromosomes, which correspond to candidate solutions. Chromosomes are usually chosen according to the quality of solutions they represent. Fitness function is allotted to every chromosome in the population. Therefore, the better the fitness of a chromosome, the possibility for selecting the chromosome is more for reproduction and the more parts of its genetic material will be passed on to the next generations.

Genetic Algorithms are easy to build and validate, which makes them highly attractive for a large number of applications. The algorithm is parallel; it can be used for large populations efficiently, even if it starts with a poor original solution it can quickly evolve to good solutions. Genetic algorithms are good at handling huge search space and navigating them, looking for optimal combination of the solutions.

Proposed Method

The flowchart of the proposed method is given in Fig. 1. The proposed method includes steps such as encoding the sequences, population, fitness function, selection, crossover, mutation and closure checking.

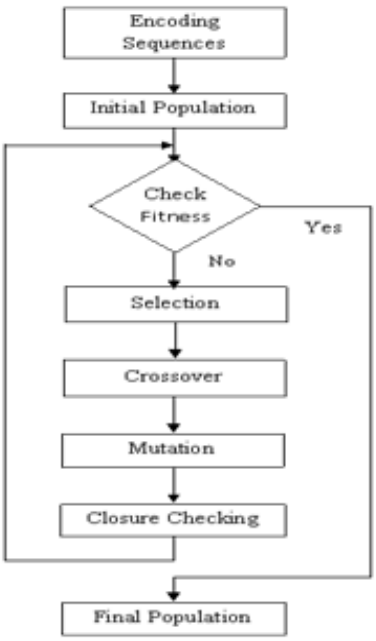

Fig. 1. Flowchart of the proposed method

a) Encoding the Sequences

There are many different approaches for encoding the sequences. We use permutation encoding approach because its format is similar to the format of the sequences. Each sequence is represented by a chromosome in genetic algorithm. We have to correctly define the chromosome to map the problems of closed sequential pattern mining.

Table 1. Encoding

Sequence	Chromosome		
	gene1	gene2	gene3
(a)(b)(cd)	a	b	cd

A sequence contains a list of elements and each element can contain one or more items. Each element of the sequence is mapped into a gene in the chromosome. Given a sequence $<e_1\ e_2\ ...\ e_n>$, it is mapped into a chromosome with n-genes. For example, a sequence (a)(b)(cd) is mapped into a 3-gene chromosome. Table 1 shows how a sequence is encoded into a chromosome.

b) Population

In general, the number of populations is fixed. We do not restrict the number of populations in our algorithm. All 1-item sequences are collected from the database and selected as the initial population. When new sequential patterns are produced during the evolvement, the new patterns are kept into next population for the selection. The new patterns are ignored if the population already contains the same patterns. We use a hash table to check whether a new pattern has already included in the population for increasing the performance.

c) Fitness Function

The candidates in the current population are evaluated using a fitness function to select the best candidates for the next generation. The fitness function is shown in "(1)". The support of a candidate α is the no of records in the database which contain α. If the candidate support is high then its fitness will be high, so that the candidate has good characteristics to consider for the next generation.

$$\text{fitness} = \begin{cases} \text{support} - \text{min\_sup} & \text{for initial population} \\ \text{fitness} \times (1 - \text{decayrate}) & \text{for remaining populations} \end{cases} \quad (1)$$

The initial value of fitness decreases during the evolvement. It shows that the candidates in the population will gradually die down to evolve. If a candidate's fitness value is less than 0 then the candidate is considered as a worthless candidate and it is not considered for the next generation.

The decay rate specifies the decreased speed of candidate's fitness. The decay rate is a percentage value between 0 and 100. If a candidate is selected by the selection process, its fitness will decrease by the speed of the decay rate. If the decay rate is high then the no of frequent patterns will be less. If the decay rate is low, such as 4%, then the no of frequent patterns will be more but it increases the runtime.

d) Selection

In every generation, the algorithm sorts all the candidates in the population in descending order based on the fitness value and then selects only the top K candidates, where K is a constant represents the number of candidates that are selected for the next generation.

e) Crossover

The crossover takes place at different points between parents with varied lengths to generate sequential patterns with varied lengths. For example, a crossover can take place at a point, which is indicated by '↕' in Table 2. Two children are created after crossover. Child1 (aq) contains the first part of parent1 and the second part of parent2. Child2 (pbc) contains the first part of parent2 and the second part of parent1.

Table 2. Crossover

Parent1	***a*↕*bc***	→	Child1	***aq***
Parent2	***p*↕*q***	→	Child2	***pbc***

If a crossover happens at the end/head of parent1 and at the head/end of parent2 then child2 will be empty. In that case, the child2 is created by reversing the child1. It is shown in Table 3. A *Crossover Rate* is employed to control the possibility of crossover when parents create their children.

Table 3. Crossover at head/end

Parent1	***abc*↕**	→	Child1	***abcpq***
Parent2	***↕pq***	→	Child2	***pqabc***

f) Mutation

Mutation is required to avoid falling all solutions in population into a local optimum. Mutation selects a random point and replaces all genes after that point with 1-item patterns. For example, the mutation changes the candidate (abc) into (apq) if p and q are 1-item patterns. Mutation Rate is a percentage that represents the likelihood of mutation when parents create their children.

g) Closure Checking

The Closure checking is used to eliminate nonclosed sequential patterns. After crossover and mutation operations a new population will be created. It is important to check whether the new population satisfies the requirements of closed sequential patterns. We call a sequence α as a closed sequential pattern if there is no super sequence of α with the same support.

h) GCSP Algorithm

Algorithm: GCSP
Input: Sequence database, min_sup, k, decay rate, crossover rate and mutation rate.
Output: The complete set of closed sequential patterns.
1: Encode the sequences.
2: Create an initial population with all 1-item sequences.
3: Calculate the fitness of the candidates.
4: Retain only candidates that have high fitness in the population.
5: Select top-k candidates in the population.
6: Perform crossover and mutation to create the new population.
7: Eliminate nonclosed sequential patterns in the new population.
8: Repeat the steps from 3 to 7 until all candidates in the population have high fitness value.

After encoding of the dataset, the GCSP algorithm starts with initial population that contains all 1-item sequences. Then the following processes are repeated until all the candidates in the population have high fitness value. The fitness value is calculated for each candidate in the population. Only the fittest candidates in the current population are retained. Top-k candidates in the current population are selected to increase the speed of crossover operation. The crossover and mutation operations are performed to create the new population. Finally closure checking is applied to eliminate the nonclosed sequential patterns.

Performance Evaluation

In our experiments we used a real world dataset BMS WebView2 of KDD CUP 2000[12] and two synthetic datasets. BMS WebView2 is a click stream data from an e-commerce web store named Gazelle and it has been used widely to assess the performance of frequent pattern mining. This dataset contains 77,512 sequences and 3340 distinct items. The average length of sequences is 4.62 items with a standard deviation of 6.07 items. The characteristics of the BMS WebView2 dataset are given in

Table 4. We generated the synthetic datasets using SPMF[13] framework. The characteristics of the two synthetic datasets are given in Table 5.

Table 4. Characteristics of the dataset

S. No.	Characteristic	Value
1	No of sequences	77512
2	No of distinct items	3340
3	Average length of sequences	4.62

Table 5. Characteristics of the Synthetic datasets

S.No.	Characteristic	Dataset1 Value	Dataset2 Value
1	No of sequences	20000	25000
2	No of distinct items	300	200
3	No of items per itemset	4	3
4	No of itemsets per sequence	5	6

All experiments were conducted on a 2GHz Intel Core2 Duo processor PC with 1GB main memory running Microsoft Windows XP. The algorithms were implemented in Java and were executed using different support values. Three sets of experiments were conducted to evaluate the performance of the GCSP algorithm. The first set compares the runtime performance of GCSP with CloSpan and ClaSP using real world dataset BMS WebView2 for different support values. The second and third sets compare the runtime performance of GCSP with CloSpan and ClaSP using synthetic datasets for different support values. The tests are based on Crossover Rate=70%, Mutation Rate=8% and Decay Rate=4%.

Fig. 2 shows the results of runtime performance using the real world dataset BMS WebView2. The X-axis is the minimum support, while the Y-axis is the algorithms runtime. The support values are set from 0.01 to 0.06. Our proposed algorithm GCSP runs faster than CloSpan and ClaSP when the support threshold is low.

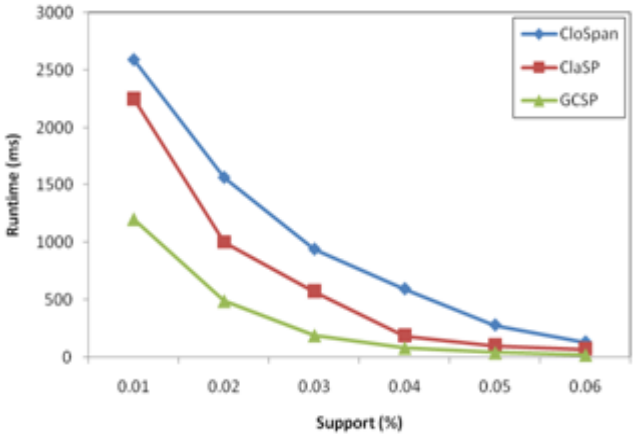

Fig 2. Performance Comparison using BMS WebView2 Dataset

Fig. 3 and Fig. 4 show the results of runtime performance using the synthetic datasets. The X-axis is the minimum support, while the Y-axis is the algorithms runtime. The support values are set from 0.01 to 0.06. Our proposed algorithm GCSP outperforms CloSpan and ClaSP on both the synthetic datasets.

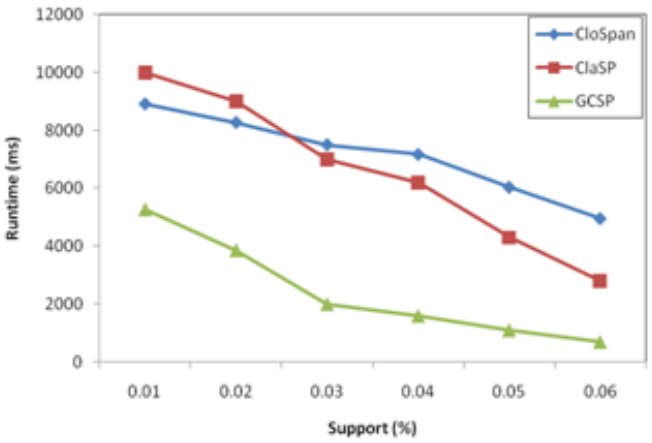

Fig 3. Performance comparison using synthetic dataset1

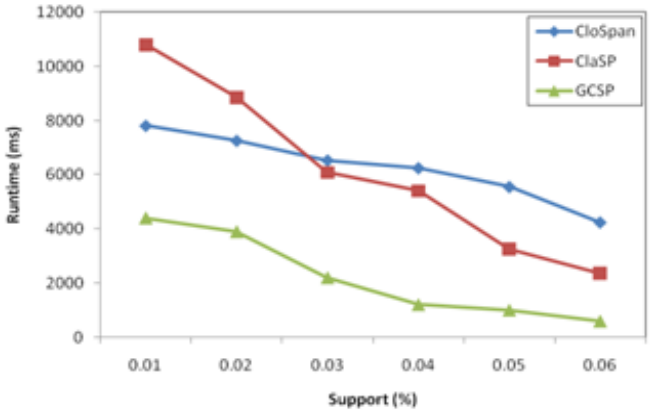

Fig 4. Performance comparison using synthetic dataset2

Conclusion

In this paper, we propose a novel algorithm GCSP to mine closed sequential patterns by using genetic approach. Closed sequential pattern mining helps users to find more interesting patterns and reduces the burden of users to explore too many patterns. Extensive experimental results on a real-world dataset and synthetic datasets show that the proposed algorithm GCSP can find closed sequential patterns efficiently and has remarkable performance compared with CloSpan and ClaSP. In our future work, we will focus on new measures for fitness function, selection and crossover to make our algorithm more efficient.

References

[1] R. Agrawal and R. Srikant, "Mining sequential patterns," Proceedings of ICDE '95, pp. 3-14, Mar. 1995.

[2] M. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," Machine Learning, vol. 42, pp. 31-60, 2001.

[3] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "PrefixSpan : Mining sequential patterns efficiently by prefix-projected pattern growth," Proc. Int'l Conf. Data Engineering (ICDE '01), pp. 215-224, Apr. 2001.

[4] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, "Sequential pattern mining using a bitmap representation," Proceedings of ACM SIGKDD '02, pp. 429-435, July 2002.

[5] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining closed sequential patterns in large databases," Proceedings of SIAM's SDM '03, pp. 166-177, May 2003.

[6] J. Wang, J. Han, and Chun Li, "Frequent closed sequence mining without candidate maintenance," IEEE TKDE., vol. 19, no. 8, pp. 1042-1056, Aug. 2007.

[7] Antonio Gomariz, Manuel Campos, Roque Marin, and Bart Goethals, "ClaSP: An efficient algorithm for mining frequent

closed sequences," PAKDD 2013, LNAI 7818, Part I, pp. 50–61, 2013.
[8] A. Bilal and A. Erhan, "An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules," Soft. Computing, vol. 10, no. 3, pp. 230–237, 2006.
[9] Sandra de Amo, Ary dos Santos Rocha, "Mining generalized sequential patterns using genetic programming," IC-AI CSREA Press, pp. 451-456, 2003.
[10] S. Ghosh, S. Biswas, D. Sarkar and P. P. Sarkar, "Mining frequent itemsets using genetic algorithm," International Journal of Artificial Intelligence & Applications, vol. 1, no.4, Oct. 2010.
[11] B. Bidgoli and William F. Punch, "Using genetic algorithms for data mining optimization in an educational web-based system, "Springer Lecture Notes in Computer Science, vol. 2724, pp. 2252-2263, 2003.
[12] Ron Kohavi, Carla E. Brodley, Brian Frasca, Llew Mason, and Zijian Zheng, "KDD-Cup 2000 organizers' report: Peeling the onion," SIGKDD Explorations, vol. 2, no. 2, pp. 86-93, Dec. 2000.
[13] Fournier-Viger P., An Open-Source Data Mining Library, http://www.philippe-fournier-viger.com/spmf/index.php?link= datasets.php, 2008, Accessed 12 Dec 2014.
[14] R.L. Haupt and S.E. Haupt, Practical genetic algorithms, Wiley, New York, 1998.
[15] M. Mitchell, Introduction to genetic algorithms, MIT Press,Cambridge,1996.