Evolving temporal association rules with genetic algorithms

MATTHEWS, Stephen G., GONGORA, Mario A. and HOPGOOD, Adrian A.

Evolving Temporal Association Rules with Genetic Algorithms

Stephen G. Matthews, Mario A. Gongora and Adrian A. Hopgood

Abstract A novel framework for mining temporal association rules by discovering itemsets with a genetic algorithm is introduced. Metaheuristics have been applied to association rule mining, we show the efficacy of extending this to another variant - temporal association rule mining. Our framework is an enhancement to existing temporal association rule mining methods as it employs a genetic algorithm to simultaneously search the rule space and temporal space. A methodology for validating the ability of the proposed framework isolates target temporal itemsets in synthetic datasets. The Iterative Rule Learning method successfully discovers these targets in datasets with varying levels of difficulty.

1 Introduction

Data Mining is the process of obtaining high level knowledge by automatically discovering information from data in the form of rules and patterns. Data mining seeks to discover knowledge that is accurate, comprehensible and interesting [9]. Association rule mining is a well established method of data mining that identifies significant correlations between items in transactional data [1]. An example of this is a rule that states “customers who purchase bread and milk also purchase cheese”. The use of such rules provides insight into transaction data to allow businesses to make better informed decisions. The usefulness of association rule mining extends to many areas ranging from biomedical and environmental to social networking and retail. With an increasing volume of information this is prevalent in time series data as well as static problems.

However, classical association rule mining assumes the dataset to be static where discovered rules are relevant across the entire dataset. In many cases this does not

Stephen G. Matthews, Mario A. Gongora and Adrian A. Hopgood
Centre for Computational Intelligence, De Montfort University, Leicester, UK e-mail: sgm@dmu.ac.uk; mgongora@dmu.ac.uk; aah@dmu.ac.uk

reflect real-world data. Often there can be a temporal pattern behind the occurrence of association rules. The scope is far reaching, many systems producing time series data have underlying processes/events that are dynamic. For example, association rules may occur more frequently in the days leading to a large sports event, or when an unforeseen event occurs, such as network intrusions. Discovering and adapting to changes with well-informed information is important in many domains, and within business it is critical for success. Association rules that incorporate temporal information have greater descriptive and inferential power [17] and can offer an additional element of interestingness.

Existing approaches to mining temporal association rules rely on identifying all itemsets that frequently occur throughout the dataset. With a large number of attributes this is computationally expensive and can lead to combinatorial explosion with classical methods. In this paper, we present an approach that incorporates a genetic algorithm [14] to mine frequent itemsets for temporal association rules without exhaustively searching the itemset space and temporal space. The temporal rules sought from the itemsets are those that occur more frequently over an interval of the dataset. They are seen as an area of greater itemset density. This research seeks to analyse the efficacy of a genetic algorithm for mining temporal association rules, where there has been very little research into this aspect of association rule mining, e.g. [5]. This is a challenging problem for a genetic algorithm because it involves searching the itemset space as well as the temporal space. Searching additional spaces other than the rule space has been shown in other variants of association rule mining such as quantitative [19, 3] to be effective. Our approach offers benefits where temporal patterns are of interest. It can be scaled up to larger problems and can include evolving additional parameters.

This paper is organised as follows. An overview of related work covering association rule mining and evolutionary computing methods, such as genetic algorithms, is discussed in Section 2. In Section 3 the genetic-algorithm based approach for mining temporal association rules is presented. An experiment to analyse the efficacy is presented and discussed with results in Section 4 and we conclude our work in Section 5.

2 Related Work

2.1 Temporal Association Rule Mining

A synopsis of preliminary work on classical association rule mining is discussed before developing the concept further to include a temporal aspect. Association rule mining is an exploratory and descriptive rule induction process of identifying significant correlations between items in boolean transaction datasets [1] used for data analysis and interpretation.

The formal definitions of association rule mining are presented as required preliminary knowledge. We assume a set of items $I = \{i_1, i_2, ..., i_M\}$ for all product items in market basket data and a set of all transactions $D = \{d_1, d_2, ..., d_N\}$. Each transaction, d_i, comprises a subset of items, referred to as an itemset, from I representing a collection of items found in a customer's shopping basket. Association rules are expressed as an implication of the form $X \Rightarrow Y$ where the consequent and antecedent are sets of boolean items where $X \cap Y = \emptyset$. For example, the rule $\{\text{bread, milk}\} \Rightarrow \{\text{cheese}\}$ implies that when bread and milk are purchased cheese is also purchased. This suggests there is a strong relationship between the sale of bread and milk, and the sale of cheese. This is the most simple form of association rules.

To extract such rules from datasets the support-confidence framework was introduced with the Apriori algorithm in [2]. Support determines the strength of a relationship by measuring how often the rule occurs in a dataset. Confidence determines how frequently the items in the consequent occur in transactions containing the antecedent, which measures the reliability of the inference. Support and confidence are defined in Equations 1 and 2 respectively.

$$\text{Support}, s(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \tag{1}$$

$$\text{Confidence}, c(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{2}$$

Minimum support and minimum confidence are introduced to limit the itemsets and rules produced to those that are significant and interesting. In the general case, a rule that occurs once in the dataset would not be considered interesting. Based on this support-confidence framework, rules are extracted by the following stages. First, the frequent itemsets are generated that have a support above the minimum support. The rules are then identified from the frequent itemsets that satisfy the minimum confidence constraint. This is a common approach for rule induction that employs a level-wise, breadth-first strategy but there are other methods that use equivalence classes, depth-first and compression techniques [21].

Association rules that have an underlying temporal behaviour can be expressed with various temporal patterns and frameworks. A key issue of classical methods that are based on the support-confidence framework is that temporal patterns having low support values can escape through the minimum support threshold. For example, consider a rule that has high support in the month of December but for the remainder of the year it is relatively much lower. This rule may not be discovered with classical association rule mining algorithms when there are rules that persistently occur throughout the dataset and consequently have higher support. Assuming that the minimum support is sufficiently low for the rule in December to be discovered, further analysis is required to ascertain any temporal property. One such property, *lifespan*, was introduced in [4] as an extension on the Apriori algorithm [2]. This is a measure of support that is relative to the lifespan

of the itemset defined by a time interval, known as temporal support. So for the example rule occurring in December, its temporal support would be relevant for a time interval representing December.

A similar problem can be found where individual items are not present throughout the entire dataset. For example, an item may be available for sale in a supermarket only during a particular seasonal period, such as British asparagus during summer. In [6, 15] the temporal element of individual items is considered rather than that of the itemsets. This variation of the problem shares the same issue with support as seen in [4] where low-support itemsets can be lost under the minimum support threshold. A related area that also focuses on the analysis of support values within a temporal framework is that of [8]. Their work introduces the concept of emerging patterns which describe itemsets as those where support increases significantly from one dataset to another. New trends can be identified by itemsets that are starting to appear more frequently.

Other methods seek to identify temporal patterns with techniques that do not directly analyse support. In [20] cyclic association rules are defined as rules that occur in user-defined intervals at regular periods throughout a dataset. For example, "*at weekends*, customers who purchase bacon and eggs also purchase sausages". This is achieved by mining association rules from partitions of the dataset and then pattern matching the rules from each partition. These are fully periodic rules because they repeatedly occur at regular intervals. Partially periodic rules [12] relax the regularity found in fully periodic so the cyclic behaviour is found in only some segments of the dataset and is not always repeated regularly. Defining the temporal intervals with calendar-based schemas is less restrictive and reduces the requirement of prior knowledge [18].

2.2 Association Rule Mining with Evolutionary Computation

Evolutionary computation is a subfield of computational intelligence that uses evolutionary systems as computational processes for solving complex problems [7]. There has been recent interest in the use of evolutionary algorithms, as well as swarm intelligence techniques, for association rule mining [19, 22, 16]. Evolutionary algorithms are metaheuristic methods that are suitable for association rule mining because they can search complex spaces and they address difficult optimisation problems. We discuss applications of evolutionary computation to association rule mining to highlight its suitability for this novel application.

With an evolutionary approach, there are generally two main strategies to encoding rules in a solution [13] that determine how rules are evolved. The Pittsburgh approach represents each individual as a ruleset and the Michigan approach encodes each individual as a single chromosome. Iterative Rule Learning is based on the Michigan approach except that the best solution is chosen after multiple runs of the genetic algorithm. The genetic cooperative-competitive approach encodes the rules in the population which collectively forms the solution.

Considerable research has focused on the use of evolutionary algorithms for mining quantitative association rules since these are present in many real-world applications. These are different from boolean association rules because they include a quantitative value describing the amount of each item. A method of mining quantitative data requires the values to be discretised into meaningful representations that are comprehensible, but this is a difficult task as the number of attributes and their parameters may not be known. Evolutionary algorithms have been shown to be capable of defining the intervals for quantitative attributes whilst simultaneously extracting association rules [19]. A recent study [3] has demonstrated the effectiveness of several genetic algorithms based on the Michigan and Iterative Rule Learning approaches for mining association rules and itemsets compared with classical mining algorithms like Apriori.

Minimum support is a very influential factor affecting the results of the mining process and it is challenging to specify a priori. Support values that are too low will yield many rules, but support values that are too high will produce too few, if any. In [22] a genetic algorithm is employed where fitness is determined based on relative confidence of association rules across the entire dataset, thus no minimum support is specified. This is shown to be suitable for both boolean and quantitative association rules. Particle swarm optimisation is an alternative method that also achieves similar results for the same purpose of not defining minimum support [16].

As well as mining quantitative data and removing the need for specifying minimum support, evolutionary computation has seen applications in only a few temporal association rule mining tasks. In [5] the Pittsburgh approach is used to mine association rules from partitions of a dataset. The resulting rules are then analysed to discover changes between partitions, similar to those of [20]. Higher-level rules are then produced from the changes in association rules to describe the underlying temporal patterns. The changes in association rules have also been evolved for the purposes of trading on the financial markets [10].

These approaches demonstrate the ability of evolutionary computation in searching for association rules and/or optimising parameters of rules (membership functions), or the induction process (support values). Our novel approach draws on the strengths of evolutionary algorithms for mining association rules that is evident from recent research. The next section describes the evolutionary algorithm approach we have adopted for mining temporal association rules.

3 Evolving Temporal Association Rules

We propose the use of a genetic algorithm to evolve temporal association rules that have high relative support over a time interval. A genetic algorithm is chosen because it is a promising solution for global search and it is capable of discovering itemsets with corresponding parameters as seen in [19, 3]. Our approach searches for itemsets occurring more frequently in an exhibition period by optimising (maximising) the relative support over a discovered time interval.

This has similarities with evolutionary methods for mining attribute intervals since we are evolving a temporal element that is also interval based.

Several test runs of the genetic algorithm were used to determine the configuration of parameters. The number of iterations of the genetic algorithm is set to 15. The genetic algorithm's population is set at 500 individuals and it is terminated at 200 generations. Elitism accounts for 1% of the new population, copy produces 25%, crossover produces 45% and mutation produces 39%. Descriptions of the genetic algorithm's configuration are now presented.

Iterative Rule Learning The Iterative Rule Learning approach is used where each chromosome represents a single itemset and the best solution from numerous runs of the genetic algorithm is selected. This approach relies on the stochastic process of genetic algorithms to yield different solutions. An advantage of Iterative Rule Learning is it produces a reduced rule set, depending on the number of iterations, that contains rules of significant temporal interest and that are easily comprehensible. In the case of classical association rule mining, with no temporal element, Iterative Rule Learning would aim to evolve a reduced rule set containing the most frequent itemsets. We do not penalise the fitness of solutions that have evolved in previous runs and so permit the same solution to be evolved. Doing so gives a clear indication of the efficacy of evolving a single isolated temporal pattern that we use as a specific target. The methodology for isolating the target is explained in more detail in Section 4.1.

Chromosome Figure 1 shows the configuration of genes in the chromosome. An integer representation encodes each item, x_i, where the ordering of items is unimportant. Lower and upper endpoints, t_0 and t_1 respectively, define the edges of the interval in which the itemset occurs most frequently. The chromosome length is fixed allowing only a specified itemset length to evolve on each run. The itemsets are evolved first because the measure (temporal support) used for identifying patterns evaluates itemsets only. The association rules are then generated from the itemsets by calculating the confidence measure after the genetic algorithm has executed.

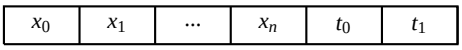

Fig. 1 Chromosome

Population Initialisation The initial population is randomly generated using the Mersenne Twister pseudorandom number generator. Setting different random seeds for each run ensures the experiment is repeatable. Upon randomly generating an item in a chromosome, it is checked against other items already generated in the same chromosome and if the item is present a new number is randomly generated until it is unique. This is repeated for each item in the chromosome to ensure all items are unique. The number of items in the dataset (e.g. inventory) must be greater than the itemset size otherwise this will result in chromosomes where

the only difference is the ordering of items. The lower and upper endpoints are randomly generated using the same method of repeating the number generation until the solution is feasible. The constraint on the endpoints is the minimum temporal support in Equation 3, this is discussed further with the fitness evaluation.

$$t_1 - t_0 >= min\_temp\_sup \tag{3}$$

Fitness Evaluation Fitness is evaluated using the relative support of the itemset over its lifespan. Equation 4 is the temporal support metric defined in [4].

$$s(X, l_X) = \frac{\sigma(X)}{l_X} \tag{4}$$

We introduce l as a time interval i.e. $l_X = [t_0, t_1]$ where t_0 is the lower endpoint and t_1 is the upper endpoint. The genetic algorithm maximises temporal support. A minimum temporal support [4] is used to prevent evolving solutions to a minimal lifespan that only cover one transaction. For example, a lifespan of 1 covers a single transaction, this produces a support of 100% for any itemset i.e. maximum fitness.

Selection Fitness proportionate selection is used to select individuals from a population for copying across to a new population or applying genetic operators. A method based on roulette wheel selection is employed. A random float value is generated between 0 and the sum of all fitness values. The fitness values are then accumulated until the accumulation is greater than the random float value. The individual selected is that which pushes the accumulation above the random number.

Genetic Operators Elitism is used to automatically copy over the best individuals from the current population to the next population without selection. A percentage of individuals are also selected and copied into the next generation.

Uniform crossover is adapted to ensure that only feasible solutions are produced, i.e. combinations of integers without duplicates. The method for crossing over only the itemsets is presented in Algorithm 1 and the stages are now briefly described. The advantage of this method is that the ordering of items remains unless a duplicate is present in the itemset.

Stage 1 (lines1 - 4) Merge the chromosomes from two selected parents into an intermediate array so that no two items from the same parent are adjacent.

Stage 2 (lines 5 - 11) Check each item in the array for duplicate values against the remaining items. If a duplicate is found the duplicate item is swapped with the next item. The result is that all duplicate items are now adjacent and the items can now be selected from the intermediate array to form an offspring.

Stage 3 (lines 12 - 18) Select items from the intermediate array by iterating over every even index value. A random integer from $[0,1]$ is added to the index and the indexed item is added to the offspring. If a 0 is generated, it is checked for duplicates with the preceding item and if a duplicate is found it adds 1 to the index otherwise it adds 0.

A random integer from $[0,1]$ determines whether the genes representing the lower and upper endpoints are copied from a single parent or they are crossed over from two parents. If they are crossed over then the feasibility of offspring is ensured by satisfying the constraint in Equation 3.

Algorithm 1 Algorithm for performing crossover on itemsets

Require: $Parent1.length \equiv Parent2.length$
for $i = 0$ to $Parent1.length - 1$ **do**
  $Auxiliary[2i] = Parent1[i]$
  $Auxiliary[2i+1] = Parent2[i]$
end for
for $i = 0$ to $Auxiliary.length - 1$ **do**
  for $j = i+2$ to $Auxiliary.length - 1$ **do**
    if $Auxiliary[i] \equiv Auxiliary[j]$ **then**
      exchange $Auxiliary[j]$ with $Auxiliary[i+1]$
    end if
  end for
end for
for $i = 0$ to $Parent1.length - 1$ **do**
  if $i > 1$ and $Auxiliary[2i-1] \equiv Auxiliary[2i]$ **then**
    $Child[i] = Auxiliary[2i+1]$
  else
    $Child[i] = Auxiliary[2i+\text{RANDOM}(0,1)]$
  end if
end for

To produce a mutated individual, a chromosome is selected and a randomly chosen gene is replaced with a randomly created value that is feasible. For the genes forming the itemset, the value must be unique and the genes for the endpoints must satisfy Equation 3.

4 Evaluation

To evaluate the efficacy of the proposed approach, several experiments have been conducted on synthetic datasets. The aim is to ascertain whether the algorithm can correctly identify areas where association rules occur more frequently.

4.1 Methodology and Datasets

The IBM Quest Synthetic Data Generator [11] [1] has been used to generate a dataset for experimentation. The generator produces datasets that replicate transactions. This approach was first used in work that focused on a retail environment [2]. A synthetic dataset is chosen rather than a real dataset so that a controlled experiment can be conducted to validate the efficacy of our approach. Individual temporal itemsets that exhibit relatively high support over an exhibition period are isolated and used as target solutions.

A dataset has been produced with the following features: 1000 transactions, 50 items, an average size of transactions of 10 and a maximal pattern length of 4. A maximal pattern cannot be part of any rule of greater length; it has no supersets that are frequent. There is no guarantee that the generated dataset contains any temporal patterns so, to include temporal information, two datasets have been augmented from the original dataset by the following process:

1. Run Apriori algorithm on dataset to produce frequent itemsets.
2. Select a frequent itemset with desired level of support.
3. Insert the itemset as a transaction near to the centre of the dataset. Transactions are constructed exclusively from the entire frequent itemset with no additional items so no unexpected correlations between items are introduced.

The itemsets with maximum support (6.8%) and midrange support (3.4%) were selected as varying levels of difficulty for the experiment. Itemsets were inserted into the dataset within bin sizes of 50 so that the lifespan of an itemset is of sufficient size for identifying temporal association rules. Figure 2 shows a histogram of the original dataset compared with augmented dataset containing the itemset $\{12, 21, 25, 45\}$ with maximum support. The horizontal axis shows the number of occurrences in bin sizes of 50. This bin size . This shows the increased occurrence of the itemset, the isolated target, that is to be discovered with the genetic algorithm. Figure 3 shows the original dataset against the other augmented dataset containing itemset $\{8, 12, 39, 45\}$ with midrange support. The peaks in these figures illustrate the more frequent occurrence of itemsets over a relatively small period of time that are target itemsets and intervals.

The itemset with midrange support (3.4%) is chosen because it is expected that this will be a more difficult dataset for the genetic algorithm. The genetic algorithm is more likely to follow local searches of itemsets because it is likely they have higher relative support values over the same lengths of time intervals. The support measure is used to evaluate fitness because this is the metric used to augment the dataset with significant temporal patterns.

[1] This is the data generator pioneered in [2] but the original link ceases to exist (http://www.almaden.ibm.com/cs/quest/syndata.html)

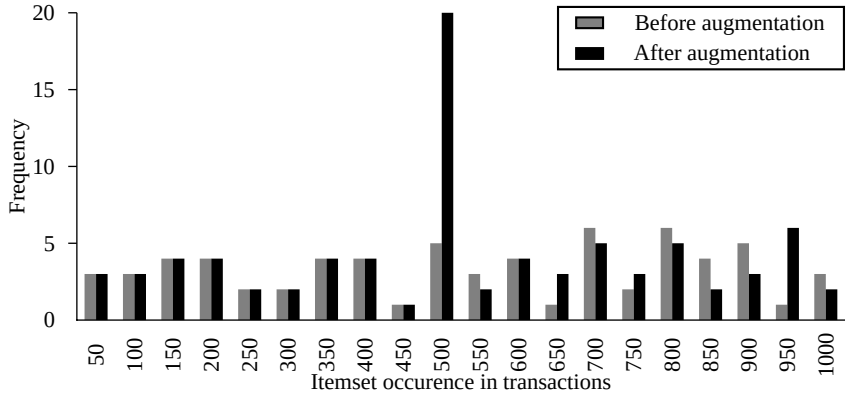

Fig. 2 Histogram of itemset {12, 21, 25, 45} with high support

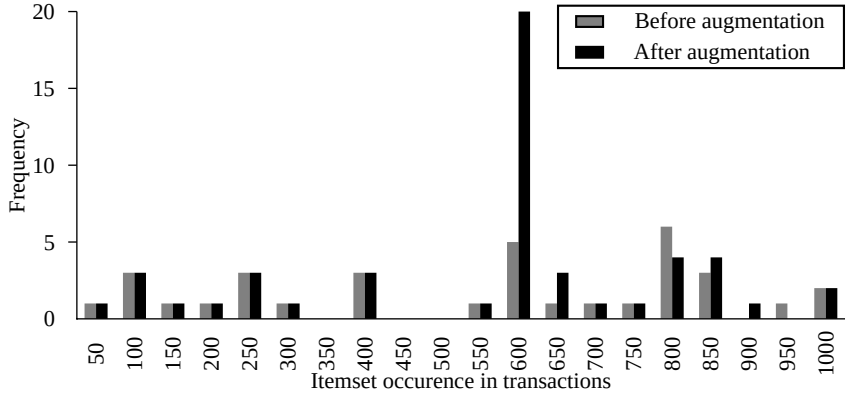

Fig. 3 Histogram of itemset {8, 12, 39, 45} with midrange support

4.2 Results

The genetic algorithm was executed 15 times with different random seeds on both augmented datasets for a maximum of 200 generations. Itemsets of length 4 were mined because this is the average maximal frequent itemset defined in the parameters of the dataset generator. The minimum temporal support was chosen based on the bin sizes used in the method for augmenting the datasets, this was set to 50. Table 1 shows the evolved itemset from each run with its corresponding interval and support values for the dataset augmented with the high support itemset. The results for this dataset show the genetic algorithm is able to consistently evolve the itemset and the endpoints for the inserted itemset in the majority of runs. The suboptimal solutions have much lower temporal support than the inserted high support itemsets. Although the termination criteria was set to 200 generations the best individuals were evolved in far fewer generations.

Seed	Itemset	Lower endpoint	Upper endpoint	Temporal Support	Generation
0	{12,21,25,45}	449	502	41.5%	51
1	{12,21,38,45}	904	960	14.3%	86
2	{12,21,25,45}	449	502	41.5%	59
3	{12,21,25,45}	449	502	41.5%	72
4	{8,12,21,43}	691	752	16.4%	37
5	{12,21,25,45}	449	502	41.5%	49
6	{12,21,25,45}	449	502	41.5%	35
7	{12,21,25,45}	449	502	41.5%	45
8	{12,21,25,45}	449	502	41.5%	60
9	{12,21,45,48}	449	502	41.5%	67
10	{12,21,38,45}	904	960	14.3%	75
11	{8,12,21,43}	687	738	15.7%	38
12	{8,12,25,45}	233	283	14.0%	20
13	{12,21,25,45}	449	502	41.5%	26
14	{12,21,25,45}	449	502	41.5%	63

Table 1 Genetic algorithm results of dataset inserted with high support itemset {12, 21, 25, 45}

The results of applying the genetic algorithm to the dataset augmented with the midrange support itemset are presented in Table 2. The results show the genetic algorithm is able to evolve the inserted itemset with the corresponding endpoints (seeds 3 and 14). However, this occurs in only a few runs of the genetic algorithm, many fewer than the previous dataset, suggesting it is a more difficult dataset. The support value across the entire dataset in Table 2 shows the genetic algorithm is more likely to evolve temporal patterns that are generally more frequent across the entire dataset. An itemset with high support occurs more frequently and so temporal patterns are found of this itemset. The histogram in Figure 4 shows an example itemset from Table 2 (seeds 4, 5, 7, 12 and 13) with high support and low temporal support (small peak in bin 800) which suggests a local optimum has evolved.

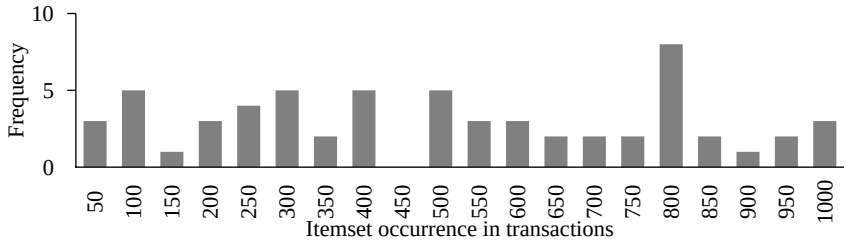

Fig. 4 Histogram of itemset {8, 12, 25, 45} in dataset augmented with midrange support itemset

From the results of executing the genetic algorithm on both datasets we can see the optimal solution is evolved. The repeatability of evolved solutions varies because of the stochastic nature of genetic algorithms but it also varies considerably between

Seed	Itemset	Lower endpoint	Upper endpoint	Temporal Support	Support	Generation
0	{12,21,38,45}	905	961	14.3%	5.7%	38
1	{1,12,21,45}	902	952	14.3%	5.0%	81
2	{8,12,21,43}	750	801	15.7%	6.1%	92
3	{8,12,39,45}	550	601	39.2%	5.1%	85
4	{8,12,25,45}	766	819	17.0%	6.2%	95
5	{8,12,25,45}	766	819	17.0%	6.2%	43
6	{8,12,21,43}	673	735	16.0%	6.1%	39
7	{8,12,25,45}	766	819	17.0%	6.2%	148
8	{8,12,21,43}	673	735	16.1%	6.1%	61
9	{10,12,21,45}	787	838	13.7%	3.5%	26
10	{12,21,38,45}	905	961	14.3%	5.7%	28
11	{8,12,21,43}	692	753	16.4%	6.1%	72
12	{8,12,25,45}	234	284	14.0%	6.2%	76
13	{8,12,25,45}	766	819	17.0%	6.2%	107
14	{8,12,39,45}	533	605	38.5%	5.1%	110

Table 2 Genetic algorithm results of dataset inserted with midrange support itemset {8, 12, 39, 45}

the two datasets. Low support items with high temporal support are more difficult to discover.

5 Conclusion

In this paper we have presented a novel approach to mining temporal association rules by discovering itemsets with a genetic algorithm. The genetic algorithm approach is capable of discovering itemsets that occur more frequently over a short time interval of a transactional dataset. The genetic algorithm method is an enhanced approach for simultaneously searching the itemset space and temporal space. The advantage of this approach is that it does not exhaustively search the dataset or require any prior partitioning.

Having identified this method to be capable, future work will include analysing its effectiveness in terms of quality of rules produced and its scalability through comparative analysis with other methods. We will investigate enhancing the fitness evaluation to reduce the chances of evolving local optima. The Iterative Rule Learning approach is a promising framework for analysing rule quality and, as already seen, individuals can be penalised to avoid searching the same areas of the fitness landscape. Further experiments on varying the number of transactions and items will provide insight into scalability. Our methodology has augmented a single temporal itemset into a synthetic dataset so future plans include using a real dataset to identify meaningful rules.

Acknowledgements This research has been supported by an EPSRC Doctoral Training Account.

References

1. Agrawal, R., Imieliński, T. and Swami, A. (1993) Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD international conference on Management of data, Washington, DC, USA, pp. 206–217.
2. Agrawal, R. and Srikant, R. (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, pp. 487–499.
3. Alcala-Fdez, J., Flugy-Pape, N., Bonarini, A. and Herrera, F. (2010) Analysis of the Effectiveness of the Genetic Algorithms based on Extraction of Association Rules. Fundamenta Informaticae, 98(1), pp. 1–14.
4. Ale, J. and Rossi, G. (2000) An approach to discovering temporal association rules. In: Proceedings of the 2000 ACM Symposium on Applied computing (SAC 00) New York, NY, USA, pp. 294–300.
5. Au, W. and Chan, K. (2002) An evolutionary approach for discovering changing patterns in historical data. In: Proceedings Of The Society Of Photo-Optical Instrumentation Engineers (SPIE), Orlando, FL, USA, pp. 398–409.
6. Chang, C.-Y., Chen, M.-S. and Lee, C.-H. (2002) Mining general temporal association rules for items with different exhibition periods. In: Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, pp. 59–66.
7. De Jong, K.A. (2006) Evolutionary computation: a unified approach. MIT Press, Cambridge, MA, USA.
8. Dong, G. and Li, J. (1999) Efficient mining of emerging patterns: discovering trends and differences. In: Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 43–52
9. Freitas, A.A. (2002) Data mining and knowledge discovery with evolutionary algorithms. Springer-Verlag.
10. Ghandar, A., Michalewicz, Z., Schmidt, M., Tô, T.-D. and Zurbrugg, R. (2009) Computational intelligence for evolving trading rules. IEEE Transactions on Evolutionary Computation, 13(1), pp. 71–86.
11. Giannella, C. (2003) IBM Quest Market-Basket Synthetic Data Generator. http://www.cs.nmsu.edu/ cgiannel/assoc_gen.html. Cited 29 May 2009
12. Han, J., Gong, W. and Yin, Y. (1998) Mining segment-wise periodic patterns in time-related databases. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, New York, NY, USA. pp. 214–218.
13. Herrera, F. (2008) Genetic fuzzy systems: taxonomy, current research trends and prospects. Evolutionary Intelligence, 1(1), pp. 27–46.
14. Holland, J.H. (1975) Adaptation in natural and articial systems. University of Michigan Press, Ann Arbor.
15. Huang, J.-W., Dai, B.-R. and Chen, M.-S. (2007) Twain: Two-end association miner with precise frequent exhibition periods. ACM Transactions on Knowledge Discovery from Data, 1(2), Article 8.
16. Kuo, R., Chao, C. and Chiu, Y. (2009) Application of particle swarm optimization to association rule mining. Applied Soft Computing, In Press, Corrected Proof.
17. Laxman, S. and Sastry, P.S. (2006) A survey of temporal data mining. Sādhanā, 31, pp. 173–198.
18. Li, Y., Ning, P., Wang, X. S. and Jajodia, S. (2003) Discovering calendar-based temporal association rules. Data & Knowledge Engineering, 44(2), pp. 193–218.
19. Mata, J., Alvarez, J. L. and Riquelme, J. C. (2002) An evolutionary algorithm to discover numeric association rules. In: Proceedings of the 2002 ACM Symposium on Applied Computing, New York, NY, USA, pp. 590–594.
20. Özden, B., Ramaswamy, S. and Silberschatz, A. (1998) Cyclic Association Rules. In: Proceedings of the Fourteenth International Conference on Data Engineering, Washington, DC, USA, pp. 412–421.

21. Tan, P.-N., Steinbach, M. and Kumar, V. (2005) Introduction to Data Mining, Addison Wesley, Boston, MA, USA.
22. Yan, X., Zhang, C. and Zhang, S. (2009) Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. Expert Systems with Applications, 36(2), pp. 3066–3076.