

Generating Market Basket Data with Temporal Information

Yingjiu Li Peng Ning † X. Sean Wang Sushil Jajodia
Center for Secure Information Systems
George Mason University, Fairfax, VA 22030
{yli, xywang, jajodia}@ise.gmu.edu
† Department of Computer Science
North Carolina State University, Raleigh, NC 27695
ning@csc.ncsu.edu

Abstract

This paper presents a synthetic data generator that outputs timestamped transactional data with embedded temporal patterns controlled by a set of input parameters. In particular, *calendar schema*, which is determined by a hierarchy of input time granularities, is used as a framework of possible temporal patterns. An example of calendar schema is (year, month, day), which provides a framework for calendar-based temporal patterns of the form $\langle d_3, d_2, d_1 \rangle$, where each d_i is either an integer or the symbol *. For example, $\langle 2000, *, 16 \rangle$ is such a pattern, which corresponds to the time intervals consisting of all the 16th days of all months in year 2000. This paper also evaluates the data generator through a series of experiments. The synthetic data generator is intended to provide support for data mining community in evaluating various aspects (especially the temporal aspects and the scalability) of data mining algorithms.

1 Introduction

Knowledge discovery in very large databases has become an important research topic in the data mining community. Although excellent ideas on new knowledge representation mechanisms and algorithms to discover such knowledge are constantly emerging, data mining researchers often face an awkward but real problem: they have no suitable data sets to examine their algorithms. One important reason is that the organizations that have collected large amount of data are usually unwilling to share their data, since by doing so they may accidentally disclose their business secret that they would rather keep to themselves. As a result, available data sets are usually not good enough to fully examine the algorithms being studied.

As an alternative to real data sets, many researchers resort to synthetic data sets that share similar features to the real ones. For example, Agrawal et al. used synthetic data sets to study the performance of several algorithms for mining association rules [AS94]. Though different from real data sets, synthetic data sets can help evaluate various aspects of the data mining algorithms, especially the scalability.

Generation of synthetic transactional data (also known as market basket data) was first introduced to study the discovery of association rules in [AS94]. However, time is not considered as a factor during the transaction generation. The generation procedure was augmented to take time into account in [RMS98], but the interaction of different time granularities is not considered and, as a result, the generation procedure may not be able to capture some characteristics that commonly exist in real data sets.

In this paper, we use *calendar schema* as a framework to incorporate temporal information in multiple granularities into synthetic market basket data. A calendar schema is determined by a hierarchy of time granularities. For example, a calendar schema can be (year, month, day). A calendar schema forms a framework for *simple calendar-based patterns* (or *calendar patterns* for short). For example, given the above calendar schema, we can derive calendar patterns such as *every day of January of 2000* and *every 16th day of January of every year*. Our synthetic data generator, which we

call *tBasket*¹, takes as input a calendar schema as well as other parameters that control the temporal and non-temporal features of the synthetic data set, and outputs a set of timestamped synthetic transactions that have some embedded temporal patterns as designated by the input parameters. The synthetic data generator is intended to provide support for data mining community in evaluating various aspects of data mining algorithms.

Considering *tBasket* as a tool for data mining research, especially for testing the scalability of temporal data mining algorithms, we hope *tBasket* is able to generate synthetic data sets with very different features as controlled by the parameters. We design a series of experiments to examine *tBasket*. In these experiments, we are particularly interested in one issue: can we generate very different data sets, especially in terms of temporal features, by varying the input parameters? We use two metrics, *itemset-index* and *pattern-index*, to examine the generated data sets in our experiments, and the results demonstrate that *tBasket* is quite effective.

The rest of the paper is organized as follows. The next section first reviews the synthetic market basket data generation proposed in [AS94], and then describes how we incorporate temporal information into the generator. Section 3 evaluates our data generator through a series of experiments. Section 4 concludes the paper.

2 The Market Basket Data Generator: *tBasket*

Our synthetic data generator is based on the well-known method discussed by Agrawal et al. in [AS94]. In the generator discussed in [AS94], time is not considered as a factor during data generation, and the generated transactions have no timestamps associated with them. In this section, we first briefly review the method proposed in [AS94], then present the representation mechanism of time by introducing the concepts of calendar schema and calendar pattern, and then discuss how to incorporate temporal information into synthetic market basket data through calendar schema and calendar patterns.

2.1 A Previous Market Basket Data Generator

The transactional data generation procedure proposed in [AS94] takes several parameters and generates a set of transactions, each of which consists of a set of items. In order to mimic the transactions in the retailing environment, it is assumed that people tend to buy sets of items together. In [AS94], such a set is called a “potentially large itemset”. The size of a potentially large itemset is chosen from a Poisson distribution with mean equal to a parameter I . Given another parameter L , totally L potentially large itemsets are generated. For each such itemset, half of its items are generated randomly, and the other half are picked from the previous itemset. This captures the phenomenon that potentially large itemsets often have common items. Each such itemset is also assigned a weight that is drawn from an exponential distribution with unit mean.

The potentially large itemsets are then used to generate D transactions, where D is another parameter. The size of each transaction is picked from a Poisson distribution with mean equal to a parameter T . Each transaction is a union of a set of potentially large itemsets that are chosen one by one according to their weights (like tossing an L -sided weighted dice). If an itemset does not fit in the transaction exactly, the remaining part of the itemset is moved to the next transaction. To capture the phenomenon that the items in a potentially large itemset are not always bought together by customers, a noise level is set from a normal distribution with mean 0.5 and variance 0.1. While adding a potentially large itemset to a transaction, we keep generating a uniformly distributed random real number between 0 and 1 and dropping an item from the itemset as long as the random number is less than the noise level.

2.2 Calendar Schema and Calendar-based Patterns

A *calendar schema* is a relational schema $R = (f_n : D_n, f_{n-1} : D_{n-1}, \dots, f_1 : D_1)$ together with a *valid* constraint (explained below). Each attribute f_i is time granularity² name like year, month, and week etc. Each domain D_i is a

¹Detailed information and software package of *tBasket* are currently available at <http://www.ise.gmu.edu/~yli/tBasket>.

²Formal definition of time granularity is given in [BJW00]. It is often convenient and sometimes necessary for users to define time granularities and then use them in calendar schemas. The reader is referred to [LMF86] and [BJW00] for generation of user-defined time granularities.

finite subset of the positive integers. The constraint *valid* is a Boolean function on $D_n \times D_{n-1} \times \dots \times D_1$ specifying which combinations of the values in $D_n \times \dots \times D_1$ are “valid”. The *valid* constraint serves two purposes. The first is to exclude the combinations that do not correspond to any time intervals due to the interaction of the calendar granularities. The second purpose of the *valid* constraint is to exclude the time intervals that we are not interested in.

Given a calendar schema $R = (f_n : D_n, f_{n-1} : D_{n-1}, \dots, f_1 : D_1)$, a *calendar-based pattern* (or *calendar pattern* for short) is a tuple on R of the form $\langle d_n, d_{n-1}, \dots, d_1 \rangle$ where each d_i is in D_i or the wild-card symbol $*$. The calendar pattern $\langle d_n, d_{n-1}, \dots, d_1 \rangle$ represents the set of time intervals that are intuitively described by “the d_1^{th} f_1 of the d_2^{th} f_2, \dots , of d_n^{th} f_n .” In the above description, if d_i is the wild-card symbol “*” (instead of an integer), then the phrase “the d_i^{th} ” is replaced by the phrase “every”. For example, given the calendar schema (week, day, hour), the calendar pattern $\langle *, 1, 10 \rangle$ means “the 10th hour on the first day (i.e., Monday) of every weeks”. Each calendar pattern in effect represents the time intervals given by a set of tuples in $D_n \times \dots \times D_1$ that are valid. For convenience, we call a calendar pattern with at least one wild-card symbol *star calendar pattern*, and a calendar pattern with no wild-card symbol *basic time interval*.

2.3 Incorporating Temporal Information

We extend the transaction data generator proposed in [AS94] by incorporating temporal information. Given calendar schema, we first generate L potentially large itemsets for each basic time interval e_0 . We call these itemsets *per-interval itemsets*. Then we generate transactions for e_0 from per-interval itemsets for the same e_0 following the exact method in [AS94]. The number of transactions for e_0 conforms to Poisson distribution with mean equal to D .

To model the phenomenon that some itemsets may be associated with temporal patterns but others may not, we require a subset (common part) of the per-interval itemsets be chosen from a single common set of itemsets and that the other part (independent part) of the per-interval itemsets be generated independently. We call the itemsets in the common set *pattern itemsets*. The pattern itemsets are shared by per-interval itemsets across some basic time intervals. We apply the same method of producing potential large itemsets in [AS94] to generate both pattern itemsets and the independent part of per-interval itemsets. We use a parameter *pattern-ratio*, P_r , to determine the percentage of the per-interval itemsets for each basic time interval that are chosen from the pattern itemsets.

So far, the remaining question is how to select the common part of per-interval itemsets from the pattern itemsets. We associate each pattern itemset with several star calendar patterns. For each basic time interval e_0 , we choose a pattern itemset as a per-interval itemset whenever one of the associated patterns covers³ e_0 . The number of calendar patterns assigned to each pattern itemset conforms to a Poisson distribution with mean N_p .

The calendar patterns assigned to pattern itemsets are picked from the space of all star calendar patterns. To model the phenomenon that the star calendar patterns covering more basic time intervals are less possible to be used in data generation than those covering fewer ones, we associate with each calendar pattern a weight, p^k , where p is a real number between 0 and 1 and k is the number of wild-card symbols in the calendar pattern (in our experiments, we use $p = 0.5$). The calendar patterns are chosen according to their weights.

To be simple, we assume the number of pattern itemsets is the same as the number of per-interval itemsets. Under such assumption, the average number of star patterns per pattern itemset, N_p , can be computed by $N_p = \frac{P_r}{p_1}$, where P_r is the pattern ratio and p_1 can be computed by

$$p_1 = \frac{(1+p)^n - 1}{\prod_1^n (|D_i| + p) - \prod_1^n |D_i|},$$

where the numerator represents the sum of weights of star calendar patterns that cover a basic time interval, and the dominator represents the sum of weights of all star calendar patterns. Note that this formula holds if all basic time intervals are valid; otherwise, it should be revised accordingly.

In summary, we list the parameters that are used in our data generator in figure 1. The upper part of table in figure 1 shows the parameters required by the original data generator proposed in [AS94], while the lower part shows the parameters related to temporal features. Figure 1 also shows the default values of the parameters.

³A calendar pattern e covers another calendar pattern e' if the set of time intervals of e' is a subset of the set of time intervals of e .

Notation	Meaning	Default value
D	Average number of transactions per basic time interval	10,000
T	Average size of the transactions	10
I	Average size of the maximal potentially large itemsets	4
L	Number of per-interval itemsets	1,000
N	Number of items	1,000
P_r	Pattern-ratio	0.4

Figure 1: Parameters for data generation

3 Evaluation of *tBasket*

3.1 Evaluation Metrics

To evaluate *tBasket*, we would like to see whether we can generate data sets with very different temporal features by adjusting the input parameters. We develop two metrics, which are closely related to the concept of association rules [AIS93], to measure the temporal features.

Let us first briefly review some concepts related to association rule and then describe the evaluation metrics. An *association rule* is a relationship between two disjoint itemsets that satisfies some user-given constraints, e.g. support and confidence rate [AIS93]. The *support* of an itemset in the set of transactions \mathcal{T} is the fraction of transactions that contain the itemset. An itemset is said to be *large* if its support in \mathcal{T} exceeds a user-given threshold *minsup*.

Given a basic time interval t under a calendar schema, we denote the set of transactions of which the timestamps are covered by t as $\mathcal{T}[t]$. For a basic time interval t in a calendar schema, an itemset is said to be *large for t* if it is large in $\mathcal{T}[t]$. For a calendar pattern e , an itemset is *large for e* if it is large for each basic time interval covered by e .

Given a support constraint, an important feature of market basket data is the number of large itemsets that are embedded in the data. Taking the temporal feature into account, we evaluate our basket data generator with the following two metrics:

1. *Itemset-index* ($\sum_k |L_k|$), where $|L_k|$ is the number of k -itemsets that are large in some star patterns. We also call $|L_k|$ *k-itemset-index*.
2. *Pattern-index* ($\sum_k |E_k|$), where $|E_k|$ is the number of star patterns in which some k -itemsets are large. We also call $|E_k|$ *k-pattern-index*.

To calculate the above two metrics, we need to find out all pairs $\langle l, e \rangle$ where the itemset l is large for the star pattern e . The itemset-index emphasizes the users' behavior from the viewpoint of large itemsets; whereas the pattern-index focuses on the aspect of temporal patterns.

As a matter of fact, if the itemsets are only large for some basic time intervals but not for any star pattern, they do not reveal much information in terms of time. Therefore, we exclude all basic time intervals from the calendar patterns in the above definition.

3.2 Synthetic Data Sets

To examine our data generator, a series of synthetic data sets are generated by varying one parameter while keeping others at their default values (see figure 1). The calendar schema R used in generation of data sets is (year:{1995-1999},month,day). In experiments, we evaluate the itemset-index and pattern-index and analyze the trend of these metrics to the change of the parameters. The size of the data sets ranges from 749 MB to 5.41 GB. The details of how to mine the large itemsets and calendar patterns efficiently, which is the foundation of computing the indices, can be found in [LNWJ01].

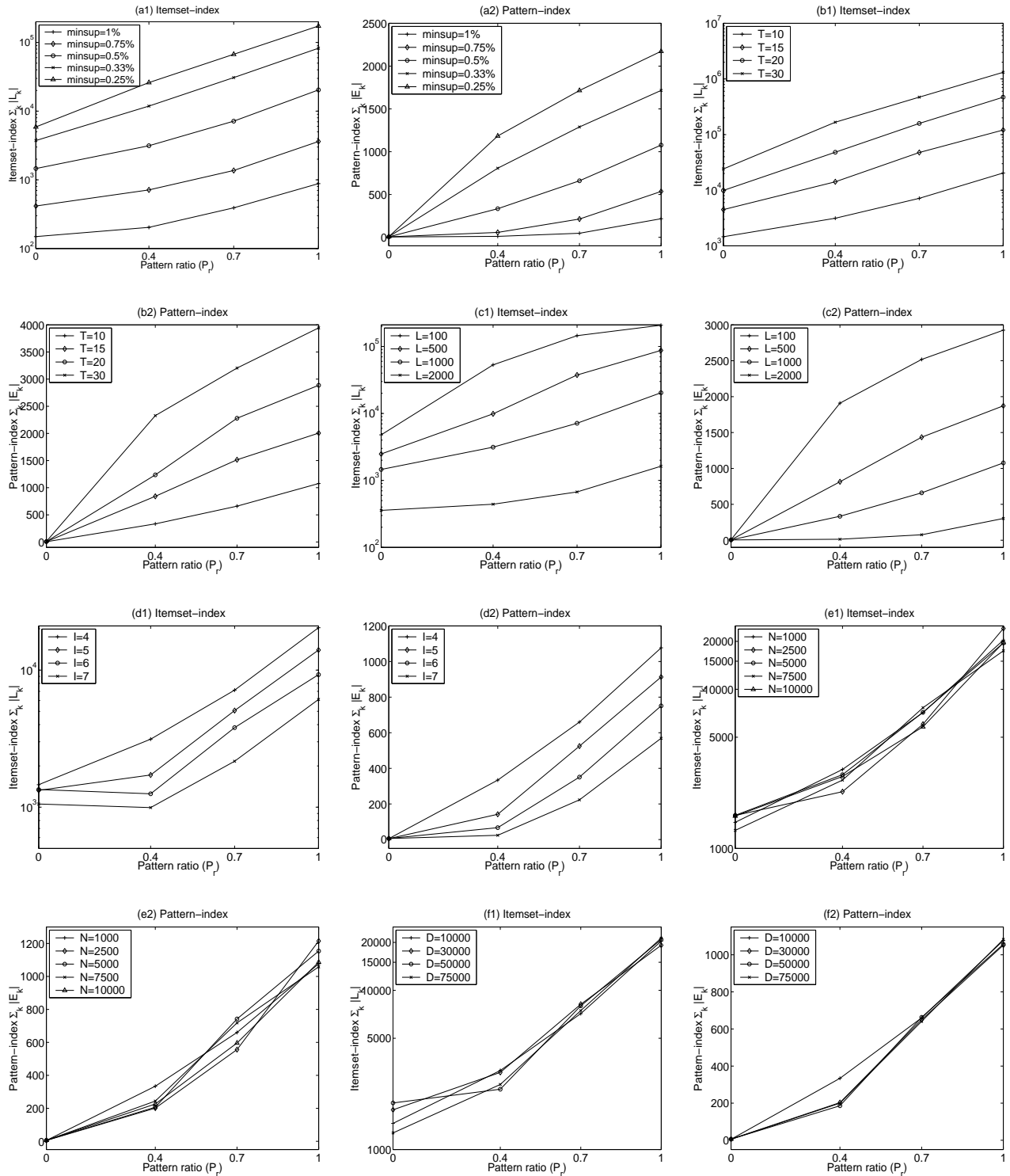


Figure 2: Evaluation result on synthetic data sets

The experiment results are shown in figure 2. The default value of $minsup$ is 0.75%. The experiment results are organized in six groups that are marked alphabetically from a to e . Each group consists of two figures, which are further denoted by natural number 1 and 2 (e.g. $a1$ and $a2$ for the group a), demonstrating the evaluation results at different pattern ratio and by varying one of the following parameters: $minsup, T, L, I, N$ and D . For each group of results, the itemset-index is always measured by a logarithm scale and the pattern-index by a linear scale.

The first observation in figure 2 is that both itemset-index and pattern-index increase as the pattern ratio becomes large. This indicates that our data generator can be effectively tuned by the pattern ratio. To illustrate this, let us examine two extreme cases of the pattern ratio. In one extreme case that the pattern ratio is zero, no pattern itemset is used to generate per-interval itemsets; in other words, the transaction set for each basic time interval is independently generated. As a result, little number of itemsets that are large for some basic time intervals are also large for some star patterns. Whereas in another extreme case that the pattern ratio is one, all per-interval itemsets are picked from the same set of pattern-itemsets. Compared with the previous case, much more itemsets are large for a larger number of star patterns.

The second observation is that the data generator is sensitive to the changes of parameters: $minsup, T, L$, and I . This observation is manifested by the distinctive separation among different curves in each of the figures 2(a-d). Therefore, a wide range of itemset-index and pattern-index can be tuned through adjusting these parameters.

The third observation is that both itemset-index and pattern-index are stable as the parameter N or D changes. This observation lies in the fact that different curves in each of figures 2(e,f) are twined together. This is a good feature in the case that users intend to scale up the number of items or the number of transactions but keep the evaluation metrics unchanged.

3.3 Real Data Set

To illustrate how to select parameters for $tBasket$ to simulate a real data set (in terms of itemset-index and pattern-index), we perform experiments to compare the result of synthetic data sets with the clicks data file in KDD Cup 2000 data sets [KB00]. The clicks data file consists of homepage request records (i.e., transactions), each of which contains attribute values describing the request and the person who sent the request. Examples of the attributes include when the request was submitted, where the person lives, and how many children the person has, and so on. The requests recorded in the clicks data file cover eight weeks (from the 6th to the 13th week in year 2000) plus 6 days (in the 14th week). We use the calendar schema $R_{KDD2K} = (week : \{6, 7, \dots, 14\}, day : \{1, 2, \dots, 7\}, timeOfDay : \{1, 2, 3\})$, where the domain values of $week$ represent the number of week of year 2000, the domain values of day represent *Sunday, Monday, ..., Saturday*, and the domain values of $timeOfDay$ represent *early morning* (0am - 8am), *daytime* (8am - 4pm), and *evening* (4pm - 12pm). The predicate $valid$ evaluates to True for all basic time intervals in the nine weeks except the last day.

We preprocess the clicks data file to remove NULL and unknown values marked with ‘?’. To simplify the problem, we focus on the categorical attributes and ignore all the attributes identified as “ignore”, “date”, “time”, and “continuous”. The preprocessed data set consists of 1,203 items and 777,480 transactions. The largest transaction consists of 100 items, the smallest transaction consists of 5 items, and the transactions contain 23.4 items on average. Using the aforementioned calendar schema R_{KDD2K} , the maximum and the minimum number of transactions per basic time interval are 27,807 and 12, respectively, and the average number is 4,180.

We mine temporal association rules in the clicks data file using $minsup = 0.75\%$. To simulate the this data set, we generate a synthetic data set by $tBasket$ using calendar schema R_{KDD2K} and parameters $T = 23, N = 1, 203, D = 4, 180$ according to the statistics of the clicks data while keeping other parameters at their default values. We also generate other two synthetic data sets by scaling up the average number of transactions per basic time interval to 5 times larger (i.e., $D = 20, 900$) and 10 times larger (i.e., $D = 41, 800$) respectively.

The experimental results are summarized in figure 3. For the case of synthetic data sets, as we observed in figure 2, both itemset-index (see figure 3(a)) and pattern-index (see figure 3(b)) increase as the pattern ratio becomes large. The figures also include the indexes for the clicks data set, whose values are independent of the pattern ratio. The intersection points between the curve for the clicks data and those for the synthetic data sets in figure 3 indicate suitable values of pattern ratio for generating the synthetic data sets that are similar to the clicks data in terms of the

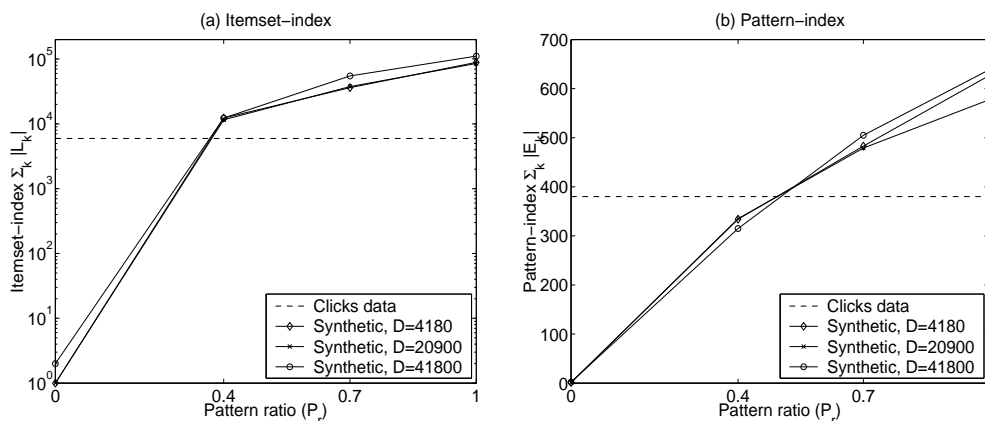


Figure 3: Relationship between clicks data and synthetic data ($minsup = 0.75\%$)

evaluation metrics.

Discussion Note that we use the same set of items when generating transactions for all basic time intervals in our experiments. To model the phenomenon that the set of items may change along the time, we can extend our generation method by using different sets of items for different basic time intervals. To further improve the ability of *tBasket* for simulating real data sets, we may further consider various types of distributions and noise factors in data generation.

4 Conclusion

In this paper, we reported the development of a synthetic data generator named *tBasket* that outputs timestamped transactional data with embedded temporal patterns controlled by a set of input parameters. In particular, calendar schema is used as a framework of possible temporal patterns. Two metrics, *itemset-index* and *pattern-index*, were developed to evaluate *tBasket*. The experiments showed that *tBasket* can generate a wide range of synthetic transactional data as controlled by the input parameters. The data generator *tBasket* was intended to provide support for data mining community in evaluating various aspects, especially the temporal aspects and the scalability, of data mining algorithms.

References

- [AIS93] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. of the 1993 Int'l Conf. on Management of Data*, pages 207–216, 1993.
- [AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of the 1994 Int'l Conf. on Very Large Data Bases*, pages 487–499, 1994.
- [BJW00] C. Bettini, S. Jajodia, and X.S. Wang. *Time granularities in databases, data mining, and temporal reasoning*. Springer-Verlag, 2000.
- [KB00] R. Kohavi and C. Brodley. 2000 knowledge discovery and data mining cup. Data for the Cup was provided by Blue Martini Software and Gazelle.com, 2000. <http://www.ecn.purdue.edu/KDDCUP/>.
- [LMF86] B. Leban, D. McDonald, and D. Foster. A representation for collections of temporal intervals. In *Proc. of AAAI-1986 5th Int'l Conf. on Artificial Intelligence*, pages 367–371, 1986.

- [LNWJ01] Y. Li, P. Ning, X. S. Wang, and S. Jajodia. Discovering calendar-based temporal association rules. In *Proc. of the 8th Int'l Symposium on Temporal Representation and Reasoning*, 2001.
- [RMS98] S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. In *Proc. of the 1998 Int'l Conf. on Very Large Data Bases*, pages 368–379, 1998.