Some Improvements on Event-Sequence Temporal Region Methods

Wei Zhang

The Boeing Company
P.O. Box 3707, MS 7L-66
Seattle, WA 98124-2207, USA

Abstract. Finding hidden temporal structures from event sequences is a difficult task, particularly when events occur irregularly over time and temporal dependencies may exist in a long time horizon. The tasks involved are not only to find event patterns represented in the form of temporal orders, but more importantly to find patterns that are described with precise *time conditions* and rules that can be applied to predict *when* a future event will occur. Recent study has shown that a new approach based on learning temporal regions is a good solution for this problem. This paper investigates this approach in a greater depth and makes several improvements. It introduces multiple rule selection methods to better uncover hidden relations. It also introduces heuristic rule pruning methods to speed up search to solve large-scale problems. Experimental results are presented which show the effectiveness of the new methods.

1 Introduction

Event sequence problems are ubiquitous in the real world. With increasing, massive amount of data being made available, solutions for such problems have become highly desired. This has led to many exciting recent developments ([Mannila *et al.*1995], [Agrawal and Srikant1995], [Srikant and Agrawal1996], [Oates *et al.*1997], [Howe and Somlo1997], and [Zhang1999]). Methods developed have been applied to a variety of problems such as telecommunication, sales transaction, and manufacturing.

Focusing on the generic event sequence problem where events are typically irregularly distributed over time, this paper investigates Zhang's (1999) temporal region approach in a great depth and makes several improvements to the existing methods. It introduces multiple rule selection methods to better uncover hidden relations. It also introduces heuristic rule pruning methods to speed up search to solve large-scale problems.

This paper is organized as follows. The next section reviews basic methods developed in the temporal region-based approach. The third section presents the methods developed for multiple rule selection and rule selection pruning. An algorithm incorporating these enhancements is also presented. After then, the paper presents an empirical study on the enhanced functionalities, followed by a conclusion pointing interesting future research.

R. López de Mántaras, E. Plaza (Eds.): ECML 2000, LNAI 1810, pp. 446–458, 2000.

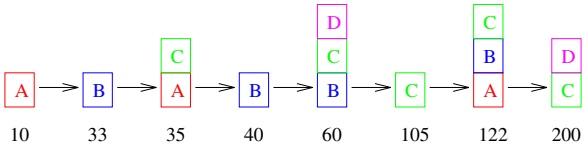

Fig. 1. An event sequence example

2 Background

2.1 The Problem

An event sequence is an ordered list of *objects* each represented with a time associated with a list of *events* that occur at the time. This means at any given time one or multiple events in different types may occur. Each event is of one (and only one) *type*. The problem is to find all significant correlations or dependency between different types of events. Such a relation could be *forward* that tells if an event of one type, say `A`, occurs now then what will be the chance an event of another type (could be the same type), say `B`, will occur in a future time, say in 5 minutes or between 5 and 10 minutes. A relation could also be *backward* that tells if an event of one type occurs now an event of another type must have occurred in a past time. When there is no time delay in a dependency of two events, we often call such a relation an *association* ([Agrawal *et al.*1996]). Sometimes there could be a mutual effect between two types of events such that one leads to the other and vice versa. Such a dependency provides a stronger association between two types of events.

The generic problem typically has the following features: The events are sparse over time, and they are irregularly distributed. Figure 1 shows a simple example how such an event sequence may present. The example contains four types of events: `A, B, C`, and `D`. Sometimes events may occur in a quite adjacent time (e.g., at time step 33 and 35). Sometimes there could be no event occurrence in a long period of time (e.g., no event between 123 and 199).

2.2 Minimal Temporal Regions

There are two basic ideas applied in the temporal region-based approach. One is rely on input event data to first develop all potential strong event correlations in the notion of *minimal temporal regions* as the hypothesis. The second is apply a set of evaluation criteria to test these hypothesis to select those of most significant correlations. Let us first look at minimal temporal regions.

A *temporal region rule* defines a temporal region condition for a *target* event type E_T, where a condition is represented in the form of a condition event type E_C associated with a period of time $[a, b]$ $(a, b \in \Re;\ 0 \leq a \leq b)$. This rule can be written as

$$E_C[a, b] \Rightarrow E_T,$$

which says two things: first, if an event of E_C occurs now then there will be *at least* one event of E_T to occur between future a and b time scope, and second, if an event of E_T occurs now then there must be *at least* one event of E_C occurred between past a and b time scope.

While there is infinite number of temporal regions that can be defined for a pair of events, our idea is to only look at rules with *minimal temporal regions* with respect to a given data set. Let S be a given sequence of events where each event records two pieces of information, the type of the event and the time this event occurs. For all pair of events of types E_C and E_T where the E_C event occurs no later than the E_T event, we can compute the *lag* between them and obtain a *set of lag values* S_{lag}. For any subset s' of S_{lag}, we can define its *minimal temporal region*, which is the smallest time interval that covers all the values in the subset, or $[min(s'), max(s')]$. The complete set of minimal temporal regions for $E_C \Rightarrow E_T$ for a given sequence is comprised of all different minimal temporal regions. Therefore, there are a total of $\binom{m+1}{2}$ minimal temporal regions for a lag set S_{lag} of size m ($m = |S_{lag}|$).

For example, for the sequence in Figure 1, the lag set for rule $C \Rightarrow A$ is $\{0, 17, 62, 87\}$. This results in 10 minimal temporal regions: [0,0], [0,17], [0,62], [0,87], [17,17], [17,62], [17,87], [62,62], [62,87], and [87,87].

2.3 Metrics

Six metrics have been developed for assessing temporal region rules.

1. **Prediction Accuracy (**AccP**):** This computes the percentage of cases that a target event occurs in the time region over all cases that a condition event occurs.

2. **Recall Accuracy (**AccR**):** This computes the same metric in the opposite temporal direction. It is the percentage of cases that a condition event occurred in the time region earlier over all cases that a target event occurs.

3. **Prediction Bonus (**BnsP**):** Sometimes target events may occur multiple times in a given time region. This metric provides a score for additional occurrences of target events. Let $FwdCnt$ be the number of cases satisfying a rule (condition-target event pair) while each condition event occurrence is only allowed to be counted at most once (this is the count used in AccR computation), and let $AllCnt$ be the count including all cases satisfying the rule where a condition event occurrence may be counted multiple times because of multiple target event occurrences. We compute the bonus as $1 - FwdCnt/AllCnt$.

4. **Recall Bonus (**BnsR**):** Similarly, we define $BwdCnt$ by counting at most once for each target event occurrence. This bonus is defined as $1 - BwdCnt/AllCnt$.

5. **Range (**Rng**):** While both AccP and AccR reward larger regions (their values increase monotonously as the size of a temporal region grows), it is important to have some metric encouraging smaller regions. Rng is the one. The BestRegionRules algorithm (see the updated version later) lets users define a lag scope in searching temporal relations, which is specified by a minimal lag `MinLag`

and a maximal lag `MaxLag`. RNG is defined as $1 - Intv(r)/(\texttt{MaxLag} - \texttt{MinLag} + 1)$, where INTV($r$) is the region size of rule r.

6. **Coverage (COV):** This metric computes the rate of cases covered by a rule over all cases that are covered by the same condition-target pair but with the full search scope defined by `MinLag` and `MaxLag`. We denote the latter as *AllCntScp*. Then COV is *AllCnt*/*AllCntScp*.

Briefly, both ACCP and BNSP assess the predicting power of events. On the other hand, both ACCR and BNSR are designed for causal analysis and diagnosis, finding reasons on event occurrences. RNG narrows down temporal regions, which is important for finding key structures of data. While sometimes this parameter may be slightly tricky to apply, fortunately, accordingly to previous study, there is often a wide range of selections available for achieving similar, good results [Zhang1999]. COV is designed for finding regions with large coverage of cases over all cases in the search scope.

3 Rule Selection Methods

3.1 Multiple Rules

The early BESTREGIONRULES algorithm has a limitation that only one region rule (the best one) may be returned for a pair of event types. To enable finding more complete set of significant relations, we add the multiple rules functionality.

The approach developed here is comprised of two steps. The first is to segment the temporal region space of a rule into several portions. After segmentation is completed, the next step is to select rules from the segmented space. We select one best rule for each segment. We also select rules across segment boundaries. This ensures the system be able to find correlations with a large lag range.

Two questions need to be answered in segmentation. First, how many segments should be selected? And second, how should we determine segments and segment boundaries? In general, answers to the first question depend on application problems. A nice feature of our approach is that since we select rules across segment boundaries, the number of segments does not affect the results on the best rules.

For the second question, several simple segmentation methods can be applied.

- **Uniform segmentation**. The simplest method is to segment a lag space uniformly. A uniform segmentation divides a lag space in the scope of `MinLag` and `MaxLag` into a number of equal-sized segments.
- **Clustering**. We can also apply clustering methods like K-MEAN to segment a space. This may improve selection of rules but certainly adds some computational cost.

A problem one needs to keep in mind when applying more sophisticated segmentation methods is that there are three different counting methods applied here: *Cnt*, *FwdCnt*, and *BwdCnt*. While *Cnt* sums over all number of the cases

covered under a region, both $FwdCnt$ and $BwdCnt$ apply a different aggregation operator—the set `Union` operation—to avoid multiple counting of an event occurrence. Segmentation using different counting methods may come up with different results. Applying which type of methods should be determined based on whether an analysis is for discovery of a forward model or a backward model and whether multiple occurrences are important.

After segmentation is finished, selection of multiple rules is trivial. We apply the following simple procedure. For each of k segments we have determined, we select the best rule in terms of the combined metric value *score*, which is a weighted sum of the six metric values. We also select a best rule across each of $k-1$ segmentation boundaries. For the ith boundary, a pre-condition event must occur in segment i and a target event must occur in segment $i+1$ or later. Furthermore, there is another parameter in the rule selection procedure. The `Minimal Score` parameter specifies that only rules with scores larger than or equal to this value are selected. Therefore, the rule selection procedure may return a maximum of $2k-1$ rules for each pair of event types.

3.2 Heuristic Pruning

The standard rule selection procedure tests over all minimal temporal regions to find a set of best region rules for each event pair. When an event sequence is very long, the number of cases under an event pair can become very large. This may result in a large set of distinctive lag values. In this situation, the minimal temporal region method no longer becomes efficient. Test over all $\binom{m+1}{2}$ minimal temporal regions is costly when the number of lag values m is very large.

To make rule selection more efficient, we introduce another parameter δ to define the *granularity* of the rule selection process. The following describes the method based on `Integer` typed lag space (It is not difficult to extend the method to the $\Re$ domain). The δ parameter defines the maximal number of lag steps that can be *skipped* in forming minimal temporal regions. When $\delta = 1$, we do not skip any lag step, so we will test all minimal temporal regions. When $\delta > 1$, lag values may be pruned. For any lag value, if the difference between its next larger lag value and its next smaller lag value is larger than δ, then this lag value must be selected. Otherwise, selection of this lag value is *optional*.

Specificly, let us look at a simple example. Let $\{0, 1, 2, 3, 4, 5, 6, 15, 25\}$ be a given lag set and let $\delta = 5$. We start the selection process from lag 0. First, we have to select 0 because there is no lag value smaller than 0. After then we consider lag 1. Selection of 1 is optional because the difference between its next larger lag 2 and previous selected lag value 0 is 2, smaller than 5. Likewise, selection of 2, 3, or 4 is all optional. Let us suppose we do not select all these. Then next we consider lag 5. At this time we have to select it because if 5 is not selected then this will cause a skip of the lag space with 6 time steps. After 5 is selected, next we have to select 6 because the difference between the next larger value 15 and the previous selected value 5 is 10, larger than 5. We can see that the selection process just went through a cluster and both boundaries of the cluster, 0 and 6, are selected. Following this, we consider 15 and 25 subsequently and

we have to choose them. Now we have obtained a pruned lag set {0,5,6,15,25}. This reduces future rule test from $\binom{10}{2}$ tests to $\binom{6}{2}$.

We develop a simple heuristic to determine whether or not to skip a lag when its selection is optional. We compare the number of cases under the current lag c_t with the number of cases under the previously selected lag c_{t-1}. If $c_t > 1.5 * c_{t-1} + 2$, then we select the current lag. Otherwise we skip it. When the number of cases under the current lag is significantly larger than the previous one, it makes sense to select this one because a temporal region either starting or ending at this point is likely to provide a better score.

Table 1. The Updated Best Region-Rules Algorithm

```
procedure BestRegionRules(S, M1, M2, W, v, k, δ)
inputs:   S                          // Event sequence
          M1                         // Minimal Lag, default = 0
          M2                         // Maximal Lag
          W                          // Weights for the score function
          v                          // Minimal Score
          k                          // The number of segments
          δ                          // The pruning factor
if (S not empty) Push(L, Pop(S))     // Move first element of S to list L
while (S not empty) do
    e := Pop(S)
    for all e' ∈ L
        let t be time of e and t' be time of e'
        let d = t − t'
        if (M1 ≤ d ≤ M2)
            let E and E' be the type of events e and e'
            AddLag(E, E', d, t')
            if (d = 0) AddLag(E', E, d, t')
    Push(L, e)
end while
SelectRules(W, v, k, M1, M2, δ)
end procedure

procedure AddLag(E_tar, E_cond, d, t)
inputs:   E_tar                      // A target event type
          E_cond                     // A condition event type
          d                          // Lag between the two events
          t                          // Start time, used as the index to represent a case
r := FindRule(E_tar, E_cond, R)      // R maintains a list of rules
if (r = NIL)                         // if this rule does not exist
    r := MakeRule(E_tar, E_cond)
    AddRule(r, R)                    // add the rule into R
SubAddLag(d, t, r)                   // add the lag and index into r
end procedure

procedure SelectRules(W, v, k, M1, M2, δ)
for all r ∈ R
    g := Segment(k, M1, M2, r)       // g defines segment boundaries
    l := PruneLagSet(r, δ)           // l is a pruned lag set
    for (i := 1, i ≤ 2k − 1, i++) best[i] := NIL  // Initialize best
    for all c ∈ pair-wise combinations of l       // c develops a region
        i := SegmentID(c, g)         // determine the segment of c
        if (Score(c, W) ≥ v and Score(c, W) ≥ Score(best[i], W)) best[i] := c
    for (i := 1, i ≤ 2k − 1, i++) PrintRegionRule(r, best[i])
end procedure
```

3.3 Algorithm

Table 1 shows the updated BestRegionRules algorithm that includes new rule selection methods. Besides taking parameters S, M_1, M_2, W, and v, the updated BestRegionRules procedure also takes the segmentation parameter k and pruning parameter δ. In SelectRules, Segment call is in the loop for all rule in the complete rule set R. This step can be moved to the outside of the loop if uniform segmentation is applied. PruneLagSet computes a set of pruned lags. After then, temporal regions are generated by pair-wise combination of the pruned lag values and then evaluated. For each temporal region, the evaluation step first determines its segment and then computes its score—the weighted sum of the six metric values using weights W. Finally, the best evaluated region for each segment is selected if its score is larger or equal to the `Minimal Score` parameter.

4 Experimental Study

We conducted a series of experiments to observe the behavior of the new methods. These experiments used the same data sets as in the [Zhang1999] work: event sequences generated based on 10 `M15-18` models (15 event types and 18 *direct* temporal relations), 10 `M30-36`, and 10 `M60-72`. Figure 2 shows one model in the `M15-18` set. Among 18 direct temporal relations, three are associations ($A_4 \Rightarrow B_2$, $B_2 \Rightarrow B_5$, and $B_5 \Rightarrow C_2$). The rest all is involved with a time lag, described with lag region $[a, b]$. Both types of relations are associated with a probability to represent the dynamics of event occurrences.

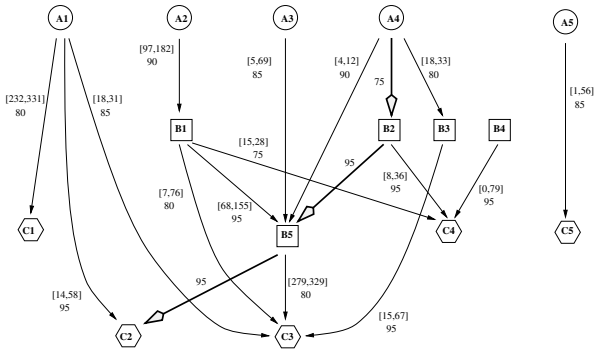

Fig. 2. An example of forward temporal models

4.1 Model Discovery

Again, this paper presents the results on forward model discovery (Similar results have been obtained on learning backward models). For forward model discovery, we do not need to apply the metrics on recall. We set weight 0 for AccR and BnsR and 1 for all other four metrics. As discussed earlier, the only slightly tricky parameter is Rng. We did some simple experiment that found setting it at 1 is reasonably good.

Also, we set the minimal-score parameter 140, the minimal lag 0 and the maximal lag 500. We set the number of segments 4. This may give a maximum of 7 region rules for each event pair. δ is set at 1 in this experiment, so no pruning is performed. For all model in the three sets, simulation was conducted with a total of 8000 events. To observe the behavior of learning, we extract learned rules when simulation is finished with 1000, 2000, 4000, and finally 8000 events respectively.

Table 2. This table gives detailed information in comparison of the learned models and the underlying models

Data Set		Match	Near Match	Region Overlap	Term Match	Unmatch	Fitness Score
M15-18 Set	Min	11.00	0.00	0.00	0.0	0.0	0.8194
	1st Qu	14.75	0.75	0.00	0.0	0.0	0.9132
	Median	17.00	1.00	1.00	0.0	0.0	0.9583
	Mean	15.70	1.30	1.00	0.0	0.0	0.9403
	3rd Qu	17.00	1.25	1.25	0.0	0.0	0.9861
	Max	18.00	4.00	3.00	0.0	0.0	1.0000
M30-36 Set	Min	29.00	0.00	0.00	0.0	0.0	0.9306
	1st Qu	32.00	1.00	0.00	0.0	0.0	0.9514
	Median	34.00	2.00	0.00	0.0	0.0	0.9757
	Mean	33.05	2.30	0.55	0.0	0.1	0.9698
	3rd Qu	34.00	3.25	1.00	0.0	0.0	0.9861
	Max	36.00	7.00	2.00	0.0	1.0	1.0000
M60-72 Set	Min	56.00	2.00	0.0	0.0	0.0	0.9306
	1st Qu	59.75	6.75	0.0	0.0	0.0	0.9514
	Median	61.50	8.50	0.0	0.0	0.0	0.9566
	Mean	62.35	8.95	0.5	0.1	0.1	0.9611
	3rd Qu	65.00	12.25	1.0	0.0	0.0	0.9757
	Max	70.00	15.00	2.0	1.0	1.0	0.9931

After learning is completed, we compared the learned models with the underlying models. For all direct temporal relation in a underlying model, we find rules in the corresponding learned model with the same names of condition event and target event. If we find a rule that has at least 80% region overlapping with this relation and vice versa (80% region of the relation is also in the rule) and the prediction accuracy of the learned rule is close to the probability of the underlying rule with difference no bigger than 0.1, then we say the two rules *match*. Otherwise, if they both have at least 50% region overlapping with each other and the probability difference is no bigger than 0.25, we say they are *"near-match"*.

Otherwise, we say the two rules *"region-overlap"* if there is at least at one point covered by the both rules. If we do not find rules that overlap with this relation then we say there is a *"term-match"* that describes a correct event temporal order. If we do not find a rule with the same condition and target names in the learned model, then we say this relation in the underlying model is *unmatch* to the learned model.

After this assessment, we computed a *fitness* function for each underlying model in comparing to its learned model. For each relation in a underlying model, we gave 1 point for a match, 0.75 point for a near-match, 0.25 point for a region-overlap, 0.1 point for a term-match, and 0 for unmatch. The fitness score is the sum of all the points divided by the number of relations in the underlying model.

Table 2 shows the performance assessed by these six measures. We separate problems from three different sets. Each set shows six statistics including `Min`, `Mean`, and `Max` on the six measures. The table concludes that learning was well performed on all three sets of problems. These statistics are made on learned rules at 4000 and 8000 event simulations. The match rate is high, in average it is about 87%. For the rest of the relations, most are near-match. The ratios for region-overlap, term-match, and unmatch are almost 0.

4.2 One Rule vs. Multiple Rules

The next experiment compares the multiple rules method (`MultiBest`) versus one using the single rule method (`OneBest`). Both procedures use the same parameter settings as described earlier. Figure 3 compares how they perform in terms of their fitness scores over simulation. It plots the median statistic plus the errorbars using the 1st Qu and 3rd Qu values. These statistics are computed over all three sets of the problems. We can see that clearly `MultiBest` outperforms `OneBest` in uncovering underlying models.

While `MultiBest` finds rules in underlying models more accurately, it also comes up with a larger set of other rules. Figure 4 compares all rules found by both procedures. Here, we use `One` and `Multi` to represent `OneBest` and `MultiBest` respectively. `True` represents the underlying model. We plot statistics on three sets of the problems separately. For `True` we plot the average number of direct relations and the average number of *transitive* relations (Transitive relations are those that can be derived from directed relations, e.g., $A \Rightarrow C$ is the transitive relation derived from $A \Rightarrow B$ and $B \Rightarrow C$). For `One` and `Multi`, we plot the average number of rules matching direct relations, transitive relations, both types of relations, and none of the relations respectively, which are denoted as `Direct`, `Transit`, `Both`, and `Others` respectively. When a rule belongs to `Both`, it is not counted in either `Direct` or `Transit`. Here, either `Match` or `Near-Match` is considered as a match.

The plot gives a general summary about the discovered rules. `Multi` performs better than `One` not only on direct relations, but more substantially on transitive relations. For transitive rules, since the implementation is set to attempt to find temporal relations in the time scope between 0 and 500 and with range under

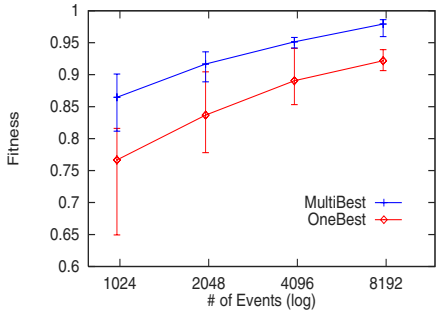

Fig. 3. `MultiBest` outperforms `OneBest` in uncovering underlying models

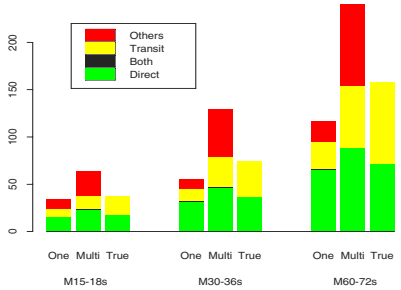

Fig. 4. A summarized comparison of the two procedures

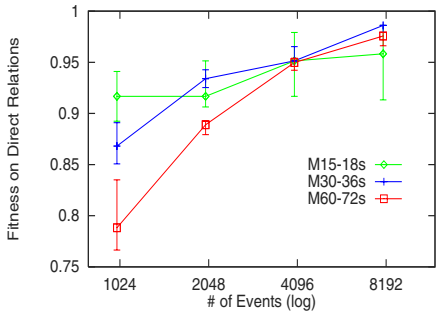

Fig. 5. Performance on finding direct relations

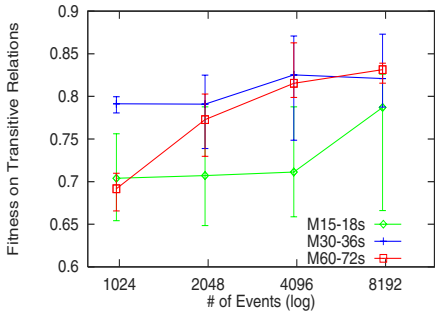

Fig. 6. Performance on finding transitive relations

100 (range is controlled by the RNG parameter), transitive relations beyond the scope are not selected. This is why the number of rules found for this part is smaller than the one in the underlying model. It is good to see that the number of rules that `Multi` selected is not substantial. Besides, the number of rules belonging to `Both` is almost 0, which means that basically direct relations and transitive relations are quite different and the algorithm can work properly to find distinctive rules for different relations.

We also observed how the multiple rules procedure performs during simulation processes. Figure 5 and Figure 6 shows the performance on both direct relations and transitive relations respectively. Again, they plots the median, 1st Qu, and 3rd Qu statistics. We compare the performance with respect to the size of models. Intuitively, a big model needs longer simulation. The plots in general suggests that for problems of size `M15-18` the most gain can be obtained by simulation to about 1000 events. The median fitness score on direct relations for `M15-18` reaches 0.92 at 1000 events. But for larger-sized problems in `M30-36` and `M60-72`, it is worth to train up to 8000 events. For these problems, the performance gain is substantial for simulation from 1000 events to 8000 events.

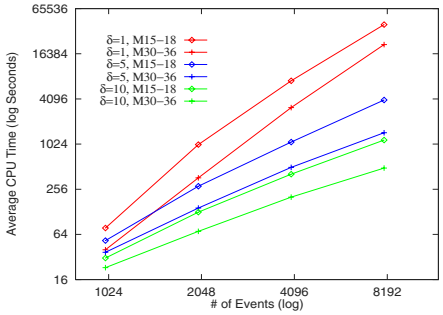

Fig. 7. CPU Time compared on with and without pruning

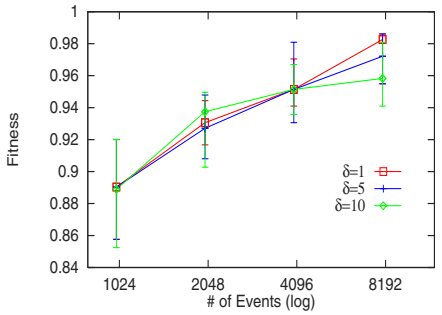

Fig. 8. With pruning the change on fitness scores is small

4.3 Speed Up

Now let us examine how the pruning factor δ affects the performance on model discovery and search time. Without surprising, we found that for the same amount of event data, in general, learning on fewer event types takes more time than on more event types. This is because with fewer event types, more cases and lags are developed for a pair of event types, thus much more temporal regions need to be evaluated. Our experiments found that on average for the same amount of data, learning for a `M15-18` problem takes about twice as much time as learning for a `M30-36` problem takes. This provides another reason to discourage longer simulation on small models. Similarly, a `M30-36` problem simulation needs about nearly twice as much time as a `M60-72` simulation with same number of events.

Accordingly, experiments for testing the δ parameter focus on smaller problems. A large model in `M60-72` probably may not generate many lags in a event pair so pruning may not be crucial. However, this does not mean for a real problem with many event types (say over several hundred types) pruning is not useful. A particular manufacturing problem in Boeing involves more than 300 types of events, but about 40% of event occurrences fall into a small set of 32 event types. In this case, we found pruning is very helpful for quick and scale-up analysis.

Figure 7 and Figure 8 plots the performance on both `M15-18` and `M30-36` problems. They compares learning without pruning ($\delta = 1$) and with pruning on two settings, $\delta = 5$ and $\delta = 10$. First, let us look at the time needed for each of the settings. Clearly, time reduction with pruning is substantial. For simulation with 8000 events on a `M15-18` model, on average it takes 40180 seconds for without pruning, but 3963 seconds with pruning on $\delta = 5$ and 1159 seconds with pruning on $\delta = 10$.

Let us now look at the performance on model discovery. We employ median, 1st Qu, and 3rd Qu on the direct-relation fitness for all problems in both sets. We can see that the performance on the three different processes is very close. Pruning does not affect the performance up to 4000 event simulation. $\delta = 5$ and

$\delta = 10$ both reach high fitness scores at about 0.95 (median) at 4000, which is the same as the un-pruned process. At 8000 events, we see $\delta = 10$ can not continue to improve the performance well, while $\delta = 5$ still keeps close to $\delta = 1$.

The plots showed clear evidence that pruning is very effective on both `M15-18` and `M30-36` problems. In conclusion, when a data set is reasonably large with respect to the number of event types, pruning is always suggested. The pruning mechanism allows this temporal region method to scale up to mine over a large amount of data.

5 Concluding Remarks

This paper investigates region-based methods for discovering temporal structures in data in a greater depth. It introduces rule selection methods to allow selection of multiple rules for more complete uncovering of hidden relations. It also introduces pruning methods for quick and scale-up analysis.

In particular, this paper showed how close direct and transitive relations can be found using the updated BestRegionRules procedure. Indeed, this procedure also discovers other relations which are neither direct nor transitive—They are *parallel* in the general sense. While all significant relations can be found under a general pool, extracting direct relations among them still remains in big challenges. In real-world applications, such analysis often relies on extensive domain knowledge. Investigating how domain knowledge should be applied and how such analysis may be formalized is a very interesting research issue.

References

Agrawal and Srikant1995. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*. IEEE Press, 1995. 446

Agrawal *et al.*1996. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthrumsamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI/MIT, 1996. 447

Howe and Somlo1997. A. Howe and G. Somlo. Modeling discrete event sequences as state transition diagrams. In *Proceedings of the Second Conference on Intelligent Data Analysis*, 1997. 446

Mannila *et al.*1995. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episode in sequences. In *Proceedings of the First International Conference of Knowledge Discovery in Databases and Data Mining*. AAAI Press, 1995. 446

Oates *et al.*1997. T. Oates, M. D. Schmill, D. Jensen, and P. R. Cohen. A family of algorithms for finding temporal structure in data. In *Proceedings of the Sixth Internationl Workshop on Artificial Intellegence and Statistics*, 1997. 446

Srikant and Agrawal1996. R. Srikant and R. Agrawal. Mining sequential patterns: generalizations and performance improvements. In *Proceedings of the Fifth International Conference on Extending Database Technology*, 1996. 446

Zhang1999. W. Zhang. A region-based learning approach to discovering temporal structures in data. In *Proceedings of the Sixteenth International Conference on Machine Learning*. Morgan Kaufmann, 1999. 446, 449, 452