

Chapter 6

Results

In this chapter, we report an extensive set of empirical evaluations. We study in detail various aspects of ApproxMAP using the evaluation method discussed in chapter 5. All experiments were run on a Dell with Dual 2GHZ Intel Xeon processors emulating 4 logical processors. The computer runs Red Hat Linux and has 2GB of memory. The program only uses one CPU in all experiments.

Unless otherwise specified, we use the version of ApproxMAP which includes the improvement discussed in section 4.6.1 but not section 4.6.2. That is, ApproxMAP uses reduced precision of the proximity matrix with k -nearest neighbor clustering.

For all experiments, we assumed that the pattern consensus sequences were the only results returned by ApproxMAP. In this section, we often refer to the consensus sequence pattern as simply *consensus pattern*. Thus, the two main parameters that affect the result are k and θ . In addition, there is an advanced parameter *min_DB_strength* that also affects the pattern consensus sequence. However, *min_DB_strength* is not meant to be changed unless it is necessary for the application. That is, *min_DB_strength* should be left at the default unless the application is specifically trying to find patterns that occur in less than 10 sequences. Thus, we assume that *min_DB_strength* is kept constant at the default, 10 sequences, for all experiments.

Furthermore, only the consensus sequences with more than one itemset in the sequence were considered as result patterns. That is all clusters with null consensus patterns (called *null clusters*) or cluster with consensus patterns that have only one itemset (called *one-itemset clusters*) were dismissed from the final result patterns because these are obviously not meaningful sequential patterns.

To accurately depict the full results of ApproxMAP we include additional information about (1) the total number of clusters, (2) the number of clusters with null consensus patterns, and (3) the number of clusters with one itemset consensus patterns. Table 6.1 gives the notations used.

Table 6.1: Notations for additional measures used for ApproxMAP

Measures	Meaning
$\ C\ $	number of clusters
$\ C_{null}\ $	number of clusters with null consensus patterns (null clusters)
$\ C_{one_itemset}\ $	number of clusters with one itemset consensus patterns (one itemset clusters)

6.1 Understanding ApproxMAP

Before we do a full evaluation of ApproxMAP using the method developed in chapter 5, we take an in depth look at some important aspects of ApproxMAP. In summary, we found that ApproxMAP was robust to the input parameter k and θ as well as the order of alignment.

6.1.1 Experiment 1.1: k in k -nearest neighbor clustering

Table 6.2: Parameters for the data generator in Experiment 1

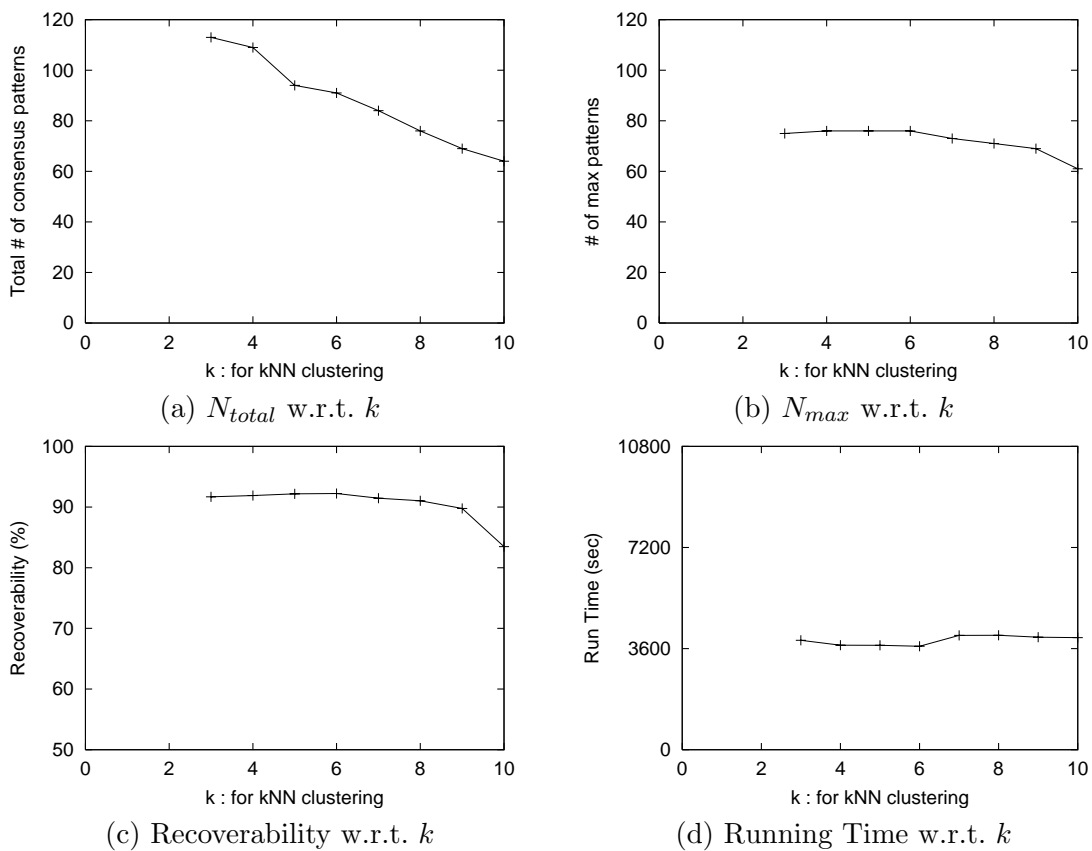
Notation	Meaning	Default value
$\ I\ $	# of items	1,000
$\ \Lambda\ $	# of potentially frequent itemsets	5,000
N_{seq}	# of data sequences	10,000
N_{pat}	# of base pattern sequences	100
L_{seq}	Avg. # of itemsets per data sequence	20
L_{pat}	Avg. # of itemsets per base pattern	14
I_{seq}	Avg. # of items per itemset in the database	2.5
I_{pat}	Avg. # of items per itemset in base patterns	2

First, we studied the influence and sensitivity of the user input parameter k . We fix other settings and vary the value of k from 3 to 10, where k is the nearest neighbor parameter in the clustering step, for a patterned dataset. The parameters of the dataset are give in Table 6.2 and the results are shown in Table 6.3 and Figure 6.1. As analyzed before, a larger value of k produces less number of clusters, which leads to less number of patterns. Hence, as expected when k increases in Figure 6.1(a), the number of consensus patterns decreases.

However most of the reduction in the consensus patterns are redundant patterns for $k = 3..9$. That is the number of max patterns are fairly stable for $k = 3..9$ at around 75 patterns (Figure 6.1(b)). Thus, the reduction in the total number of consensus patterns returned does not have much effect on recoverability (Figure 6.1(c)). When k is too large though ($k = 10$), there is a noticeable reduction in the number of max patterns from 69 to 61 (Figure 6.1(b)). This causes loss of some weak base patterns and thus the recoverability decreases somewhat as shown in Figure 6.1(c). Figure 6.1(c) demonstrates that there is a wide range of k that give comparable results. In this experiment, the recoverability is sustained with no change in precision for a range of $k = 3..9$. In short, ApproxMAP is fairly robust to k . This is a typical property of density based clustering algorithms.

Table 6.3: Results for k

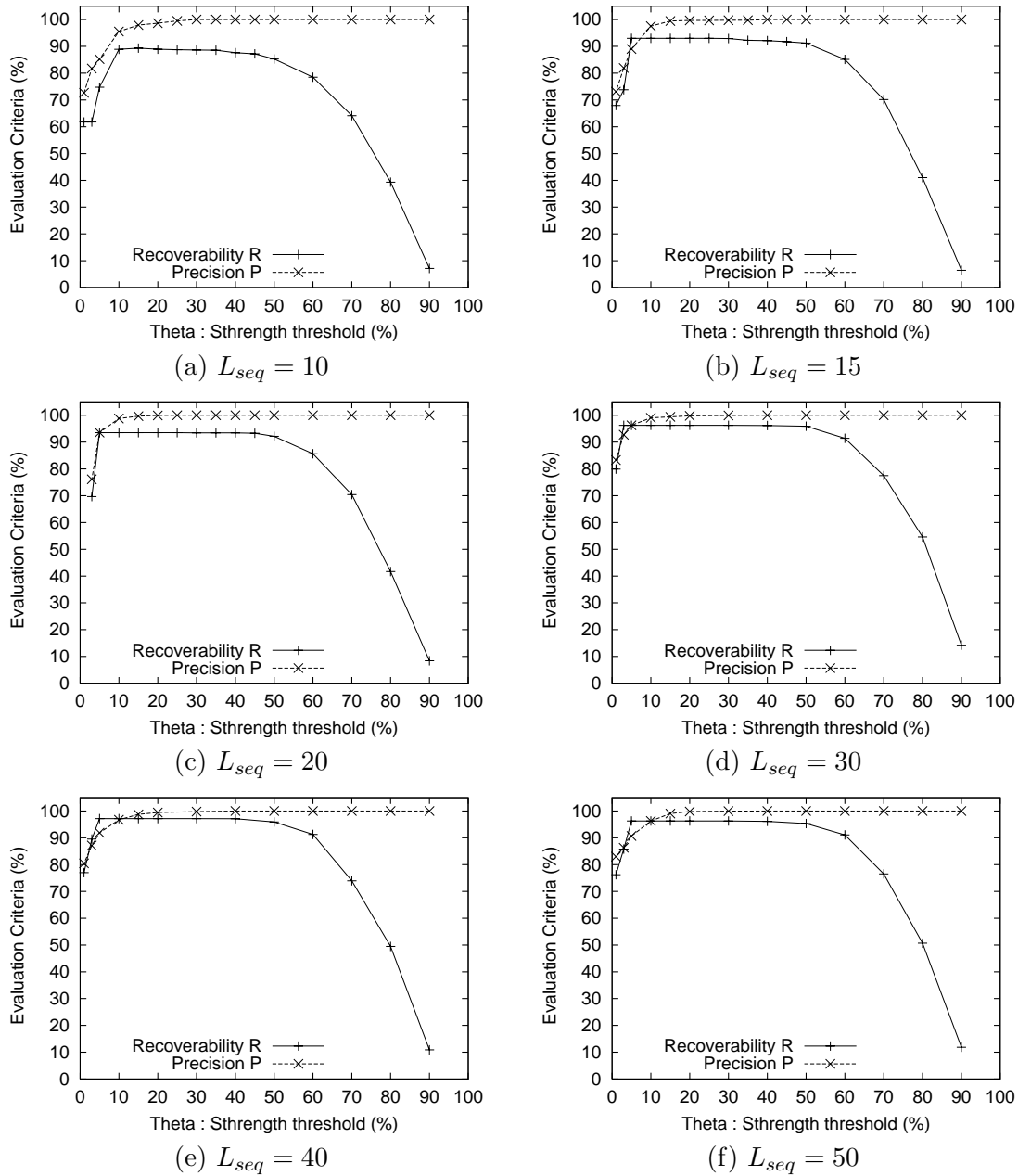
k	Recoverability	Precision	N_{spur}	N_{redun}	N_{total}	N_{max}	Running Time(s)
3	91.69%	100.00%	0	38	113	75	3898
4	91.90%	100.00%	0	33	109	76	3721
5	92.17%	100.00%	0	18	94	76	3718
6	92.23%	100.00%	0	15	91	76	3685
7	91.45%	100.00%	0	11	84	73	4070
8	91.04%	100.00%	0	5	76	71	4073
9	89.77%	100.00%	0	0	69	69	4007
10	83.47%	100.00%	0	3	64	61	3991

Figure 6.1: Effects of k 

In terms of running time, Figure 6.1(d) indicates that the performance of ApproxMAP is not sensitive to parameter k . It is stable at a little over an hour for all $k = 3..10$.

6.1.2 Experiment 1.2: The Strength Threshold

In ApproxMAP, we use two strength cutoff points, θ and δ , to filter out noise from weighted sequences. Here, we study the strength threshold to determine its properties empirically.

Figure 6.2: Effects of θ 

Since both cutoff points have the same property, we use θ as the representative strength cutoff parameter. We ran 6 experiments on 6 different patterned datasets. The patterned data was generated with the same parameters given in Table 6.2 except for L_{seq} and L_{pat} . We experimented on varying $L_{seq} = 10..50$ with $L_{pat} = 0.7 \cdot L_{seq}$.

We then studied the change in recoverability and precision as θ is changed for each dataset. Without a doubt, in Figure 6.2 the general trend is the same in all datasets.

In all datasets, as θ is decreased from 90%, recoverability increases quickly until it levels

off at $\theta = 50\%$. We see that precision stays high at close to 100% until θ becomes quite small. Clearly, when $\theta = 50\%$, ApproxMAP is able to recover most of the items from the base pattern without picking up any extraneous items. That means that items with strength greater than 50% are all pattern items. Thus, as a conservative estimate, we set the default value for pattern consensus sequence at 50%. In all our experiments, the default value of $\theta = 50\%$ is used unless otherwise specified.

On the other side, when θ is too low precision starts to drop. We further see that in conjunction with the drop in precision, there is a point at which recoverability drops again. This is because, when θ is too low the noise is not properly filtered out. As a result too many extraneous items are picked up. This in turn has two effects. By definition, precision is decreased. Even more damaging, the consensus patterns with more than half extraneous items now become spurious patterns and do not count toward recoverability. This results in the drop in recoverability. In the dataset with $L_{seq} = 10$ this occurs at $\theta \leq 10\%$. In all other datasets, this occurs when $\theta \leq 5\%$.

The drop in precision starts to occur when $\theta < 30\%$ for the dataset with $L_{seq} = 10$. In the datasets of longer sequences, the drop in precision starts near $\theta = 20\%$. This indicates that items with *strength* $\geq 30\%$ are probably items in the base patterns.

Moreover, in all datasets, when $\theta \leq 10\%$, we see a steep drop in precision. This indicates, that many extraneous items are picked up when $\theta \leq 10\%$. Since recoverability is close to 100% when $\theta = 10\%$, this means that most of the items with strength less than 10% are extraneous items. Hence, as a modest estimation, we set the default value for the variation consensus sequence at 20%. This modest estimate will allow ApproxMAP to detect almost all pattern items while picking up only a small number of extraneous items in the variation consensus sequence.

In summary, this experiment indicates that 20%-50% is in fact a good range for the strength threshold for a wide range of datasets.

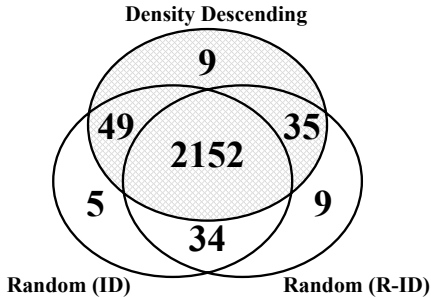
6.1.3 Experiment 1.3: The Order in Multiple Alignment

Now, we study the sensitivity of the multiple alignment results to the order of sequences in the alignment. We compare the mining results using the density-descending order, density-ascending order, and two random orders (sequence-id ascending and descending order) for the same dataset specified in Table 6.2. As expected, although the exact alignment changes slightly depending on the orders, it has very limited effect on the consensus patterns.

The result shows that (Table 6.4), all four orders generated the exact same number of patterns that were very similar to each other. The number of pattern items detected that were identical in all four orders, column $N_{commonI}$, was 2107. In addition, each order found an additional 100 to 138 pattern items. Most of these additional items were found by more than one order. Therefore, the recoverability is basically identical at 92%.

Table 6.4: Results for different ordering

Order	Recoverability	N_{extraI}	N_{patI}	$N_{commonI}$	Precision	N_{spur}	N_{redun}	N_{total}
Descending Density	92.17%	0	2245	2107	100.00%	0	18	94
Ascending Density	91.78%	0	2207	2107	100.00%	0	18	94
Random (ID)	92.37%	0	2240	2107	100.00%	0	18	94
Random (Reverse ID)	92.35%	0	2230	2107	100.00%	0	18	94

Figure 6.3: Comparison of pattern items found for different ordering

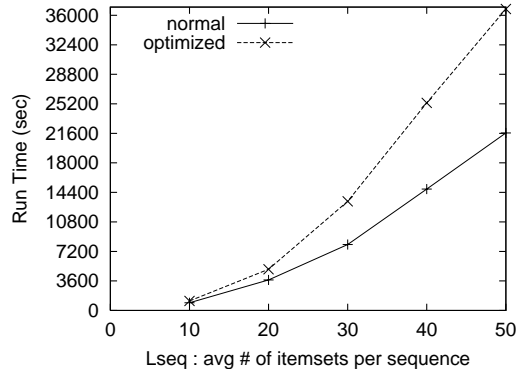
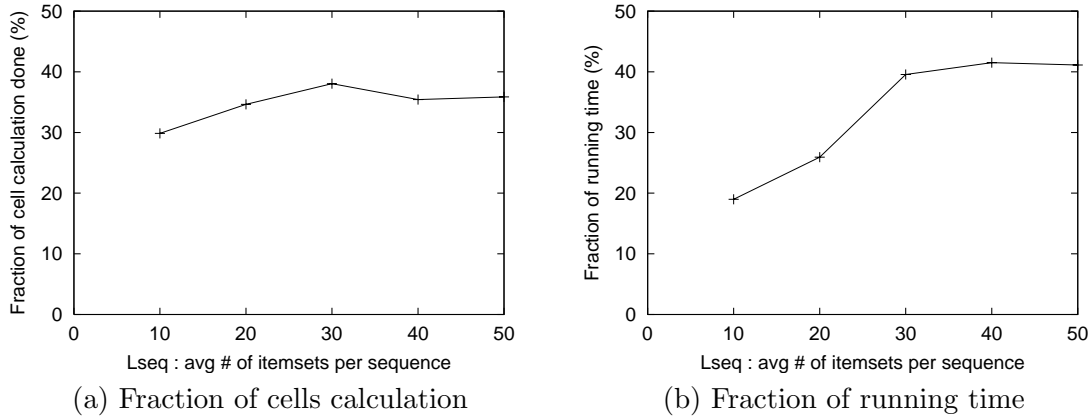
While aligning patterns in density descending order tends to improve the alignment quality (the number of pattern items found, N_{patI} , is highest for density descending order at 2245 while lowest for density ascending order at 2207), ApproxMAP itself is robust with respect to alignment orders. In fact, the two random ordering we tried gave comparable number of pattern items as the density descending order.

Figure 6.3 gives a detailed comparison of the pattern items detected by the two random orders and the density descending order. 2152 pattern items detected were identical in the three orders. 9 pattern items were detected by only the density descending order. 2201 (or 2187) pattern items were detected by both the density descending order and a random order. Essentially, a random order detected about 98% ($49/2245=2\%=35/2245$) of the pattern items detected by the density descending order plus a few more pattern items (roughly $40 \simeq 34+5 \simeq 34+9$) not detected by the density descending order.

6.1.4 Experiment 1.4: Reduced Precision of the Proximity Matrix

As discussed in section 4.5, the straight forward ApproxMAP algorithm has time complexity $O(N_{seq}^2 \cdot L_{seq}^2 \cdot I_{seq})$. It can be optimized with respect to $O(L_{seq}^2)$ by calculating the proximity matrix used for clustering to only the needed precision. This was discussed in section 4.6.1. Here we study the speedup gained empirically. Figure 6.4 shows the speedup gained by using the optimization with respect to L_{seq} in comparison to the vanilla algorithm. We see that such optimization can reduce the running time to almost linear with respect to L_{seq} .

To investigate further the performance of the optimization, we looked at the actual number of cell calculations saved by the optimization. That is, with the optimization, the modified

Figure 6.4: Running Time w.r.t. L_{seq} **Figure 6.5: Fraction of calculation and running time due to optimization**

proximity matrix has mostly values of ∞ because $k \ll N$. For those $dist(seq_i, seq_j) = \infty$, we looked at the dynamic programming calculation for $dist(seq_i, seq_j)$ to see how many cells in the recurrence table could be skipped. To understand the savings in time, we report the following in Figure 6.5(a).

$$\frac{\sum \text{the number of cells in the recurrence table skipped}}{\sum \text{the total number of cells in the recurrence table}} \cdot 100\%$$

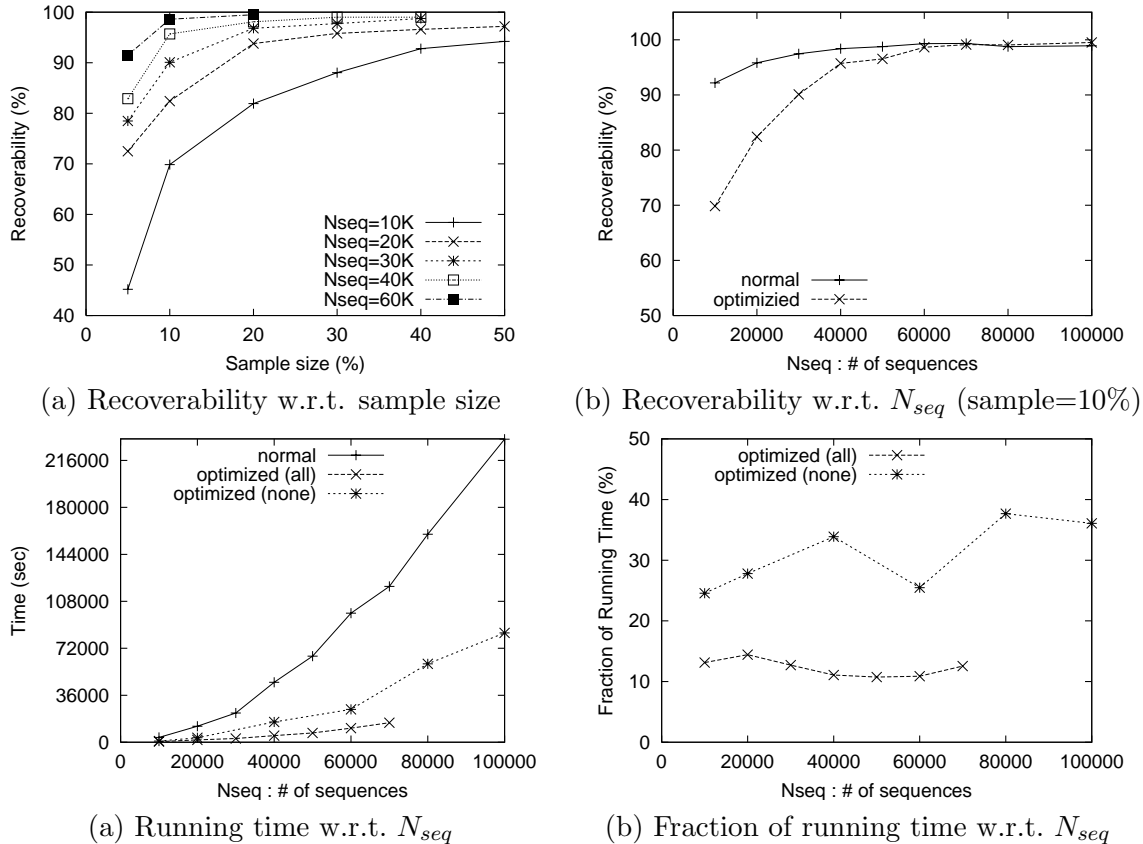
We see that when $10 \leq L_{seq} \leq 30$, as L_{seq} gets larger more and more proportion of the recurrence table calculation can be skipped. Then at $L_{seq} = 30$, the proportion of savings levels off at around 35%-40%.

This is directly reflected in the savings in running time in Figure 6.5(b). Figure 6.5(b) reports the reduction in running time due to optimization as a proportion of the original running time. The proportion of savings in run time increases until $L_{seq} = 30$. At $L_{seq} = 30$ it levels off at around 40%. Thus, we expect that when $L_{seq} \geq 30$, the optimization will give a 40% reduction in running time. This is a substantial speedup without any loss in the

accuracy of the results.

6.1.5 Experiment 1.5: Sample Based Iterative Clustering

Figure 6.6: Sample based iterative clustering



In section 4.6.2 we discussed how to speedup the clustering step by using a sample based iterative partitioning method. Such change can optimize the time complexity with respect to $O(N_{seq}^2)$ at the cost of some reduction in accuracy and larger memory requirements. Obviously, the larger the sample size the better the accuracy with slightly less gain in running time. In this section, we study the tradeoff empirically to determine the appropriate sample size. The experiments also suggests when such optimizations should be used.

Figure 6.6(a) presents recoverability with respect to sample size for a wide range of datasets. All runs use the default $k' = 3$. We see that when $N_{seq} \geq 40,000$, recoverability levels off at 10% sample size. When $N_{seq} < 40,000$, recoverability levels off at a larger sample size. However, when $N_{seq} = 40,000$ it takes less then 13 hours even without the optimization. Thus, the experiment suggests that the optimization should be used for datasets when $N_{seq} \geq 40,000$ with sample size 10%. For datasets with $N_{seq} < 40,000$, as seen in Figure 6.6(b), sample size 10% is too small. Thus, a larger sample size should be used if the

normal algorithm is not fast enough for the application. The experiments indicate that the sample size should have at least 4000 sequences to get comparable results. As expected, in the smaller datasets with $N_{seq} < 40,000$, the running time is fast even when using a larger sample size. When $N_{seq} = 30,000$ and sample size=20% the running time was only 90 minutes.

Figure 6.6(b) and (c) show the gain in running time and the loss in recoverability with respect to N_{seq} with the optimization (sample size=10%, $k' = 3$). *optimized (all)* is a simple hash table implementation with all proximity values stored and *optimized (none)* is the implementation with none of the proximity values stored. The results clearly show that the optimization can speedup time considerably at the cost of negligible reduction in accuracy. Figure 6.6(d) show that the optimization can reduce running time to roughly 10%-40% depending on the given conditions.

6.2 Experiment 2: Effectiveness of ApproxMAP

In this section, we study the effectiveness of ApproxMAP using the full evaluation method discussed in chapter 5 on some manageable sized datasets. We ran many experiments with various synthetic data sets. The trend is clear and consistent. The evaluation results reveal that ApproxMAP returns a succinct but accurate summary of the base patterns with few redundant or spurious patterns. It is also robust to both noise and outliers in the data.

6.2.1 Experiment 2.1: Spurious patterns in random data

In this section, we study empirically under what condition spurious patterns are generated from completely random data ($\|\mathcal{I}\| = 100, N_{seq} = 1000, L_{seq} = 10, I_{seq} = 2.5$). For random data, because there are no base patterns embedded in the data, evaluation criteria recoverability, precision, and number of redundant patterns do not apply. The only important evaluation measure is the number of spurious patterns, N_{spur} , generated by the algorithm. We include the total the number of result patterns returned, N_{total} , for completeness. Since there are no base patterns in the data $N_{total} = N_{spur}$.

Recall that there are two parameters, k and θ in ApproxMAP. We first study the cutoff parameter θ . To determine the threshold at which spurious patterns will be generated, T_{spur} , we ran 8 experiments varying θ for a particular k . Each individual experiment has a constant k from 3 to 10.

Here we first report the full results of the experiment with the default value $k = 5$. When $k = 5$, there are 90 clusters of which only 31 clusters had 10 or more sequences. That means that the remaining 59 clusters will not return a consensus pattern no matter how low the cutoff is set because $min_DB_strength = 10$ sequences. Recall that when generating pattern consensus sequences, the greater of the two cutoff, θ or $min_DB_strength$ is used. Therefore, the following theorem shows that the lowest value of θ that can introduce the most spurious

Table 6.5: Results from random data ($k = 5$)

θ	N_{total}	N_{spur}	$\ C\ $	$\ C_{null}\ $	$\ C_{one_itemset}\ $
50%	0	0	90	90	0
40%	0	0	90	89	1
30%	0	0	90	89	1
20%	0	0	90	86	4
16%	1	1	90	85	4

Table 6.6: Full results from random data ($k = 5$)

Cluster ID	Cluster size	$22\% \geq \theta \geq 19\%$	$18\% \geq \theta \geq 17\%$	$16\% \geq \theta \geq 0$
$Cluster_1$	42	$\langle(A)\rangle$	$\langle(A)\rangle$	$\langle(A)\rangle$
$Cluster_2$	61	$\langle(B)\rangle$	$\langle(B)(E)\rangle$	$\langle(B)(E)(B)\rangle$
$Cluster_3$	22	$\langle(C)\rangle$	$\langle(C)\rangle$	$\langle(C)\rangle$
$Cluster_4$	50	$\langle(D)\rangle$	$\langle(D)\rangle$	$\langle(D)\rangle$
$Cluster_5$	60			$\langle(F\ G)\rangle$
$\ C_{one_itemset}\ $		4	3	4
N_{spur}		0	1	1
$\ C_{null}\ $		86	86	85
$\ C\ $		90	90	90

patterns is 16%.

THEOREM 4. Given a weighed sequence, $wseq = \langle WX_1 : v_1, \dots, WX_l : v_l \rangle : n$, when $min_DB_strength$ is kept constant, the pattern consensus sequences generated by ApproxMAP stay constant regardless of θ for $0 \leq \theta \leq \frac{min_DB_strength}{n}$.

Proof: According to Algorithm 2, when generating pattern consensus sequences the actual cutoff applied is the greater of the two specified cutoffs, θ or $\frac{min_DB_strength}{n}$. Thus, the cutoff applied is $\frac{min_DB_strength}{n}$ for $0 \leq \theta \leq \frac{min_DB_strength}{n}$, regardless of θ .

By Theorem 4, when θ is lowered enough such that it is smaller than $\frac{min_DB_strength}{cluster_size}$ for all clusters, there is no change in the results regardless of θ . That is the result is constant for $0 \leq \theta \leq \frac{min_DB_strength}{size(C_{max})}$ where $size(C_{max})$ is the number of sequences in the biggest cluster.

In this experiment the biggest cluster had 61 sequences. Thus, all results are the same for $0 \leq \theta = 16\%$ ($\theta \leq \frac{min_DB_strength}{61} = \frac{10}{61} = 16.4\%$). Hence the smallest possible effective value of θ is 16%.

The results are given for $16\% \leq \theta \leq 50\%$ in Table 6.5. Noting that the first spurious pattern occurs when $\theta = 16\%$ we investigated the region $16\% \leq \theta < 20\%$ in more detail. The results for $0\% \leq \theta \leq 22\%$ are given in Table 6.6. It includes the actual consensus sequences generated for all non null clusters (the first five rows). The next two rows summarize how many of them are one itemset clusters and how many are spurious patterns. It is then followed by the number of null clusters in the result. These three add to the total number of clusters given next.

Table 6.7: Results from random data ($\theta = 50\%$)

k	N_{total}	N_{spur}	$\ C\ $	$\ C_{null}\ $	$\ C_{one_itemset}\ $
3	0	0	134	134	0
4	0	0	103	103	0
5	0	0	90	90	0
6	0	0	86	84	2
7	0	0	57	56	1
8	0	0	61	61	0
9	0	0	59	59	0
10	0	0	56	55	1

Table 6.8: Results from random data at $\theta = T_{spur}$

k	T_{spur}	N_{total}	N_{spur}	$\ C\ $	$\ C_{null}\ $	$\ C_{one_itemset}\ $
3	26%	1	1	134	132	1
4	19%	1	1	103	99	3
5	18%	1	1	90	86	3
6	27%	1	1	86	81	4
7	18%	1	1	57	47	9
8	23%	2	2	61	55	4
9	27%	1	1	59	54	4
10	26%	1	1	56	53	2

We see that up to $\theta = 19\%$ there are no spurious patterns returned and only four one-itemset consensus sequences returned. At $\theta = 18\%$ the first spurious pattern is generated when an additional itemset (E) is introduced in the consensus sequence from $Cluster_2$. Thus, the point at which the first spurious pattern occurs, T_{spur} , is 18% for $k = 5$. Then at $\theta = 16\%$ two more itemsets are introduced. First, one more itemset, (B), is introduced again to the consensus sequence from $Cluster_2$. Second $Cluster_5$, a null cluster up to this point, now has an itemset in its consensus sequence. The new consensus sequence $\langle\langle F G \rangle\rangle$ is a one-itemset consensus sequence. By Theorem 4, for all $\theta < 16\%$, the result is the same as that when $\theta = 16\%$

In summary, when $k = 5$, there is no spurious pattern generated for $\theta > 18\%$ and only one spurious pattern for $\theta \leq 18\%$. The spurious pattern is $\langle\langle B \rangle\rangle$ for $17\% \leq \theta \leq 18\%$ and $\langle\langle B \rangle\rangle$ for $0\% \leq \theta \leq 16\%$.

The summary of all 8 experiments varying k from 3 to 10 is given in Tables 6.7 and 6.8. Table 6.7 give the results for different k while keeping θ at the default value 50%. All experiments had no clusters with a meaningful (more than one itemset) consensus sequence. In fact, almost all clusters resulted in null consensus sequences. Five of eight experiments had all clusters with null consensus sequences. The other three experiments, k equal to 6, 7 and 10, gave only 2, 1, and 1 cluster with a one-itemset consensus sequence respectively. All of these one-itemset clusters, were small and had only one random item aligned. The exact

cluster sizes that had a random item align to produce a one-itemset consensus sequence were 30, 31, 20 and 19 sequences. All other clusters could not align any items at all. These good results are not surprising when we look at each experiment in detail. In the eight experiments, the cluster sizes ranged from 1 to 153 sequences with many being tiny. In total, 63% had less than 10 sequences. These clusters could not generate any consensus sequences. This is discussed more later.

In Table 6.8, we report the threshold, T_{spur} , along with the other evaluation measures for each k . The threshold at which the first spurious pattern occurs ranges from 18% to 27%.

Clearly, ApproxMAP generates no spurious patterns from the random data set ($|\mathcal{I}| = 100, N_{seq} = 1000, L_{seq} = 10, I_{seq} = 2.5$) for a wide range of k ($3 \leq k \leq 10$) when $\theta > 27\%$. This is in line with our study of θ in section 6.1.2 that it is very unlikely that extraneous items appear in the consensus pattern when $\theta \geq 30\%$

Discussion : ApproxMAP and random data

ApproxMAP handles random data very well. Although the algorithm generated many clusters (all experiments generated from 56 to 134 clusters), when $\theta > 27\%$ all the clusters produced pattern consensus sequences with either 0 or 1 itemset which were easily dismissed.

There are two reasons why there is no pattern consensus sequence for a cluster. First, if the cluster is tiny it is not likely to have much item weights greater than *min_DB_strength*. Recall that the default value of *min_DB_strength* is 10 sequences. Thus, clusters with less than 10 sequences had no pattern consensus sequence.

Second, not enough sequences in the cluster could be aligned to generate any meaningful consensus itemsets. That is in all our experiments at most one itemset could be aligned over all the itemsets. A one itemset sequence is clearly, not a meaningful sequential pattern. This is not surprising since the probability of two long sequences being similar by chance is quite small. Thus, it is very unlikely that enough sequences will align to produce patterns simply by chance.

6.2.2 Experiment 2.2: Baseline study of patterned data

The full parameters of the patterned database is given in Table 6.9. We optimized the parameters k and θ by running a set of experiments.

The first step was to find a reasonable θ that would work for a range of k . So for the first experiment, we varied k from 3 to 10 with θ set at the default value of 50%. The results, given in Table 6.10, showed that recoverability was not high enough in the region where the number of redundant patterns were small ($N_{redun} < 5$ which would be $k \geq 5$).

Therefore, we ran a second experiment with a lower $\theta = 30\%$ while varying k . As seen in the results given in Table 6.11 in general recoverability was more acceptable with good precision. So we used $\theta = 30\%$ to determine the optimal k .

Table 6.9: Parameters for the data generator for patterned data

Notation	Meaning	Default value
$\ I\ $	# of items	100
$\ \Lambda\ $	# of potentially frequent itemsets	500
N_{seq}	# of data sequences	1000
N_{pat}	# of base pattern sequences	10
L_{seq}	Avg. # of itemsets per data sequence	10
L_{pat}	Avg. # of itemsets per base pattern	7
I_{seq}	Avg. # of items per itemset in the database	2.5
I_{pat}	Avg. # of items per itemset in base patterns	2

Table 6.10: Results from patterned data ($\theta = 50\%$)

k	recoverability	Precision	N_{total}	N_{spur}	N_{redun}
3	91.85%	1/169=99.41%	15	0	7
4	88.75%	1/143=99.30%	13	0	6
5	88.69%	0/128=100.00%	11	0	4
6	87.45%	0/96=100.00%	8	0	1
7	78.18%	0/91=100.00%	8	0	2
8	82.86%	0/85=100.00%	7	0	1
9	81.09%	0/84=100.00%	7	0	1
10	64.40%	0/71=100.00%	6	0	1

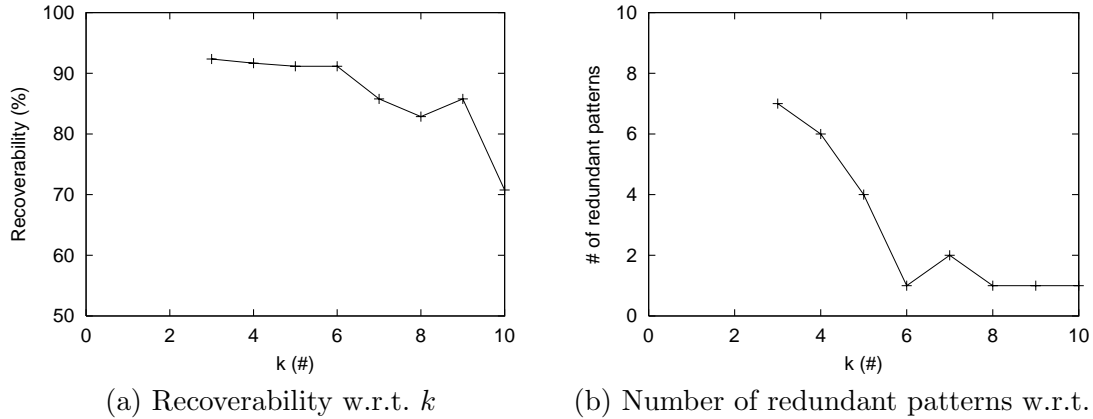
As expected, as k increase, more sequences are clumped together to form less clusters. This is reflected in the reduction of N_{total} . Initially the clusters merged in this process are those with similar sequences built from the same base pattern. This can be seen for $k = 3..6$ where N_{redun} decreases from 7 to 1 and little change occur in recoverability from 92.36% to 91.16%. However, when k is increased beyond 6, small clusters (sequences built from less frequent base patterns) are merged together and we start to loss the less frequent patterns resulting in decreased recoverability. For $k = 6..10$, we see this phenomena where recoverability is decreased from 91.16% to 70.76%. Figure 6.7(a) depicts the drop in recoverability at $k = 6$. Figure 6.7(b) illustrates that the number of redundant patterns level off at the same point ($k = 6$). Hence, $k = 6$ is the optimal resolution for clustering this dataset.

Now, we wanted to optimize θ for the optimal $k = 6$. Thus, we ran an experiment with $k = 6$ and varied θ from 20% to 50% (Table 6.12). For the optimal value, depending on the application, a user would choose either (1) the smallest θ with no extraneous items ($\theta = 40\%$) or (2) the largest θ with good recoverability and precision even though this point could include some extraneous items. We choose to go with the later option because the definition for extraneous items is the most conservative possible. That is, not all extraneous items are truly extraneous. See section 5.4 for details.

For this dataset, we decided that the precision=84.8% for $\theta = 20\%$ was too low. The region $25\% \geq \theta \geq 35\%$ had both good recoverability and precision. Since $\theta = 25\%$ had the

Table 6.11: Results from patterned data ($\theta = 30\%$)

k	recoverability	Precision	N_{total}	N_{spur}	N_{redun}
3	92.36%	1-5/179=97.21%	15	0	7
4	91.66%	1-2/153=98.69%	13	0	6
5	91.16%	1-4/136=97.06%	11	0	4
6	91.16%	1-3/106=97.17%	8	0	1
7	85.77%	1-1/100=99.00%	8	0	2
8	82.86%	1-4/90=95.56%	7	0	1
9	85.77%	1-0/90=100.00%	7	0	1
10	70.76%	1-4/82=95.12%	6	0	1

Figure 6.7: Effect of k , the nearest neighbor parameter.

same number of pattern items found with $\theta = 30\%$ but two more extraneous items, $\theta = 25\%$ was dropped from consideration.

Now, let us compare the results from $\theta = 30\%$ and $\theta = 35\%$. Although the recoverability are the same, the actual number of pattern items found are different. The number of pattern items found are 103 and 100 respectively. There are two circumstances in which there is no change in recoverability even though more pattern items are found.

First, an increase in the pattern items for a non-max pattern has no affect on recoverability unless the non-max pattern becomes longer than the original max pattern because only the pattern items found in the max pattern is used to calculate recoverability.

Second, even if the additional pattern items found are from the max pattern, if $\max_{\{P_j(i)\}} \|B_i \otimes P_j\|$ is already greater than $E(L_{B_i}) \cdot \|B_i\|$, the recoverability does not change. This is because when the expected length of the base pattern, $E(L_B) \cdot \|B_i\|$ is smaller than the observed value $\max_{\{P_j(i)\}} \|B_i \otimes P_j\|$, we truncate $\frac{\max_{\{P_j(i)\}} \|B_i \otimes P_j\|}{E(L_{B_i}) \cdot \|B_i\|}$ to 1.

To be precise, the optimal point is the one where the most pattern items are found with the least possible extraneous items. Thus, after debating between $\theta = 35\%$ and $\theta = 30\%$, we decided that $\theta = 30\%$ was the optimal point ¹.

¹The difference between the results produced by $\theta = 35\%$ and $\theta = 30\%$ are only 4(=106-102) items. One(=3-2) of them is an extraneous item and the other three are pattern items. All the pattern consensus

Table 6.12: Results from patterned data ($k = 6$)

θ	Recoverability	N_{item}	N_{patI}	N_{extraI}	Precision	N_{total}	N_{spur}	N_{redun}
20%	93.10%	125	106	19	84.80%	8	0	1
25%	91.16%	108	103	5	95.37%	8	0	1
30%	91.16%	106	103	3	97.17%	8	0	1
35%	91.16%	102	100	2	98.04%	8	0	1
40%	88.69%	98	98	0	100.00%	8	0	1
45%	88.69%	97	97	0	100.00%	8	0	1
50%	87.45%	96	96	0	100.00%	8	0	1

Table 6.13: Parameters for ApproxMAP

k	# of neighbor sequences	6
θ	the strength cutoff for pattern consensus sequences	28%..30%
δ	the strength cutoff for variation consensus sequences	20%
$min_DB_strength$	the optional parameter for pattern consensus sequences	10 sequences
$max_DB_strength$	the optional parameter for variation consensus sequences	10%

By investigating the region $25\% < \theta < 35\%$ in more detail we found that the consensus sequences are exactly the same in the region $28\% \geq \theta \geq 30\%$. That is the optimal results can be obtained with $k = 6$ and $28\% \geq \theta \geq 30\%$ for this dataset.

Now let us take a closer look at the mining result from the optimal run ($k = 6, \theta = 30\%$). Under such settings, ApproxMAP finds 8 pattern consensus sequences. The patterns and their variations are given along with the base pattern used to build the dataset in Table 6.14. For variation consensus sequences, we use the default settings as given in Table 6.13. Note that we have 9 clusters but only 8 consensus patterns because there is one null cluster ($PatConSeq_9$).

As shown in Table 6.14, each of the 8 pattern consensus sequences match a base pattern well. There were no spurious patterns. The pattern consensus sequences do not cover the three weakest base patterns ($BaseP_8, BaseP_9$, and $BaseP_{10}$). The recoverability is still quite good at 91.16%. In general, the pattern consensus sequences recover major parts of the base patterns with high expected frequency in the database.

The pattern consensus sequences cannot recover the complete base patterns (all items in the base patterns) because, during the data generation, only parts of base patterns are embedded into a sequence. Hence, some items in a particular base pattern may have much lower frequency than the other items in the same base pattern in the data. When we explored the weighted sequences by trying different cutoffs for θ , we saw that the full weighted sequences

sequences for $\theta = 30\%$ is given in Table 6.14. As θ changes from 35% to 30% an additional extraneous item, 58 in the fourth itemset in $PatConSeqs$, is picked up. The missed pattern items when $\theta = 35\%$ are (1) the last item, 51, in $PatConSeq_3$, (2) and the two items in the last itemset, (2 74), in $PatConSeqs$. The first does not affect recoverability because $PatConSeq_3$ is a redundant pattern. The second does not affect recoverability because the expected length of $PatConSeq_8$ is $0.6 * 13 = 7.8$. Thus, finding 11 items of the 13 items in the base pattern is the same as finding all 13 items when calculating recoverability.

Table 6.14: Consensus sequences and the Base Patterns for $k = 6$ and $\theta = 30\%$

BaseP _i (E(F _g):E(L _g))	P	Pattern <100: 85: 70: 50: 35: 20>
BaseP ₁ (0.21:0.66)	14	<(15 16 17 66) (15) (58 99) (2 74) (31 76) (66) (62) (93) >
PatConSeq ₁	13	<(15 16 17 66) (15) (58 99) (2 74) (31 76) (66) (62) >
VarConSeq ₁	18	<(15 16 17 66) (15 22) (58 99) (2 74) (24 31 76) (24 66) (50 62) (83) >
BaseP ₂ (0.161:0.83)	22	<(22 50 66) (16) (29 99) (94) (45 67) (12 28 36) (50) (96) (51) (66) (2 22 58) (63 74 99) >
PatConSeq ₂	19	<(22 50 66) (16) (29 99) (94) (45 67) (12 28 36) (50) (96) (51) (66) (2 22 58) >
VarConSeq ₂	25	<(22 50 66) (16) (29 99) (22 58 94) (2 45 58 67) (12 28 36) (2 50) (24 96) (51) (66) (2 22 58) >
PatConSeq ₃	15	<(22 50 66) (16) (29 99) (94) (45 67) (12 28 36) (50) (96) (51) >
VarConSeq ₃	15	<(22 50 66) (16) (29 99) (94) (45 67) (12 28 36) (50) (96) (51) >
BaseP ₃ (0.141:0.82)	14	<(22) (22) (58) (2 16 24 63) (24 65 93) (6) (11 15 74) >
PatConSeq ₄	11	<(22) (22) (58) (2 16 24 63) (24 65 93) (6) >
VarConSeq ₄	13	<(22) (22) (22) (58) (2 16 24 63) (2 24 65 93) (6 50) >
BaseP ₄ (0.131:0.90)	15	<(31 76) (58 66) (16 22 30) (16) (50 62 66) (2 16 24 63) >
PatConSeq ₅	11	<(31 76) (58 66) (16 22 30) (16) (50 62 66) >
VarConSeq ₅	11	<(31 76) (58 66) (16 22 30) (16) (50 62 66) (16 24) >
BaseP ₅ (0.123:0.81)	14	<(43) (2 28 73) (96) (95) (2 74) (5) (2) (24 63) (20) (93) >
PatConSeq ₆	13	<(43) (2 28 73) (96) (95) (2 74) (5) (2) (24 63) (20) >
VarConSeq ₆	16	<(22 43) (2 28 73) (58 96) (95) (2 74) (5) (2 66) (24 63) (20) >
BaseP ₆ (0.121:0.77)	9	<(63) (16) (2 22) (24) (22 50 66) (50) >
PatConSeq ₇	8	<(63) (16) (2 22) (24) (22 50 66) >
VarConSeq ₇	9	<(63) (16) (2 22) (24) (22 50 66) >
BaseP ₇ (0.054:0.60)	13	<(70) (58 66) (22) (74) (22 41) (2 74) (31 76) (2 74) >
PatConSeq ₈	16	<(70) (58) (22 58 66) (22 58) (74) (22 41) (2 74) (31 76) (2 74) >
VarConSeq ₈	18	<(70) (58 66) (22 58 66) (22 58) (74) (22 41) (2 22 66 74) (31 76) (2 74) >
PatConSeq ₉	0	cluster size was only 5 sequences so no pattern consensus sequence was produced
VarConSeq ₉	8	<(70) (58 66) (74) (74) (22 41) (74) >
BaseP ₈ (0.014:0.91)	17	<(20 22 23 96) (50) (51 63) (58) (16) (2 22) (50) (23 26 36) (10 74) >
BaseP ₉ (0.038:0.78)	7	<(88) (24 58 78) (22) (58) (96) >
BaseP ₁₀ (0.008:0.66)	17	<(16) (2 23 74 88) (24 63) (20 96) (91) (40 62) (15) (40) (29 40 99) >

given from ApproxMAP accurately stores this information. The less frequent items in the base patterns were in the weighted sequence but with smaller weights. This is illustrated by the lighter (weaker) items in the longer variation consensus sequences. For example, *VarConSeq₁* has 5 additional items (22, 24, 24, 50, and 93) compared to *PatConSeq₁*. Four of them are extraneous items (22, 24, 24, and 50) while the last one (93) is not. It comes from the base pattern *BaseP₁*, just with less frequency compared to the other items in the base pattern. These weaker items are not included in the pattern consensus sequences because their item strengths are not frequent enough to clearly differentiate them from extraneous items.

Precision is excellent at $\mathcal{P} = 1 - \frac{3}{106} = 97.17\%$. Thus, clearly all the pattern consensus sequences are highly shared by sequences in the database. In all the pattern consensus sequences, there is only three items (the lightly colored items 58, 22, and 58 in the first part of *PatConSeq₈*) that do not appear on the corresponding position in the base pattern. These items are not random items injected by the algorithm, but rather repeated items, which clearly come from the base pattern *BaseP₇*. These items are still classified as extraneous items because the evaluation method uses the most conservative definition. See section 5.4 for a detailed discussion on this issue.

It is interesting to note that a base pattern may be recovered by multiple pattern consensus sequences. For example, ApproxMAP forms two clusters whose pattern consensus sequences approximate base pattern *BaseP₂*. This is because *BaseP₂* is long (the actual length of the base pattern is 22 items and the expected length of the pattern in a data sequence is 18 items) and has a high expected frequency (16.1%). Therefore, many data sequences in the database are generated using *BaseP₂* as a template. As discussed in chapter 5, sequences are generated by removing various parts of the base pattern and combining with other items. Thus, two sequences using the same long base pattern as the template are not necessarily similar to each other. As a result, the sequences generated from a long base pattern can be partitioned into multiple clusters by ApproxMAP. One cluster with sequences that have almost all of the 22 items from *BaseP₂* (*PatConSeq₂*) and another cluster with sequences that are shorter (*PatConSeq₃*). The one which shares less with the base pattern, *PatConSeq₃*, is classified as a redundant pattern in our evaluation method ($N_{redun} = 1$).

Based on the above analysis, we can see that ApproxMAP provides a succinct summary of the database. That is, ApproxMAP is able to summarize the 1000 sequences into 16 consensus sequences (two for each partition) both accurately and succinctly. The 16 pattern consensus sequences resemble the base patterns that generated the sequences very well (recoverability=91.16%, precision=97.17%). No trivial or irrelevant pattern is returned ($N_{total} = 8, N_{spur} = 0, N_{redun} = 1$).

Note that in real applications, there is no way to know what the true underlying patterns are. Thus, the approach used in this section to find the optimal parameter setting can not be utilized. However, consistent with results in section 6.1, this section clearly shows that

Table 6.15: Effects of noise ($k = 6, \theta = 30\%$)

$1 - \alpha$	running time	recoverability	Precision	N_{total}	N_{spur}	N_{redun}
0%	14	91.16%	1-3/106=97.17%	8	0	1
10%	16	91.16%	1-1/104=99.04%	9	0	2
20%	10	91.16%	1-1/134=99.25%	12	0	5
30%	10	90.95%	1-0/107=100.00%	9	0	2
40%	16	88.21%	1-0/95=100.00%	9	0	2
50%	10	64.03%	1-0/68=100.00%	8	0	3

ApproxMAP is robust with respect to the input parameters. That is many settings give results comparable to the optimal solution ($k = 5$ or $6; 25\% \geq \theta \geq 35\%$). More importantly, for a wide range of k and θ the results are at least a sub optimal solution. For $k=3$ to 9 and $25\% \geq \theta \geq 50\%$, all results had recoverability greater than 82.98% and precision greater than 95.56%.

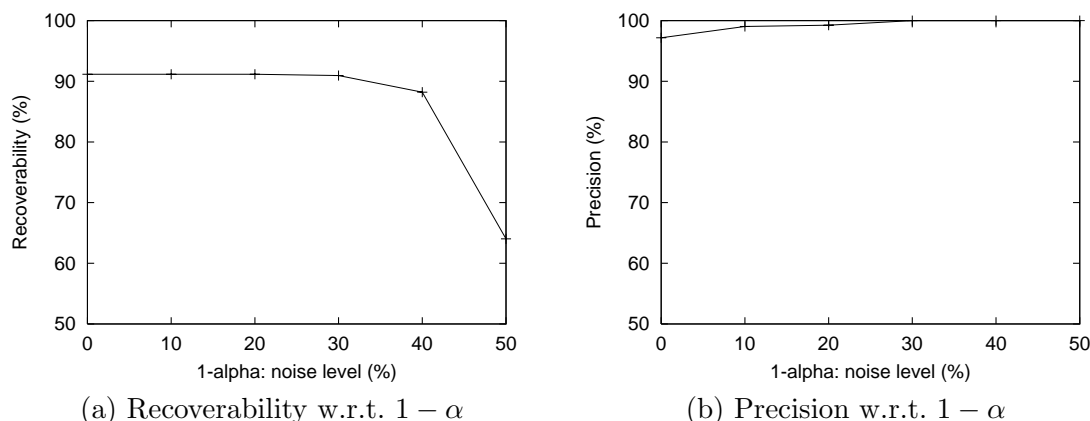
The 16 consensus sequences give a good overview of the 1000 sequences. Thus, in a real application, a careful scan of the manageable sized results can give a domain expert good estimates (even with a sub optimal setting) about what possible patterns they can search for in the data. This is really what makes **ApproxMAP** a powerful exploratory data analysis tool for sequential data. Pattern search methods (sometimes called pattern matching) are much more efficient than pattern detection methods. Users can use pattern search methods to confirm and/or tune suggested patterns found from **ApproxMAP**.

6.2.3 Experiment 2.3: Robustness with respect to noise

Here we evaluate the robustness of the **ApproxMAP** with respect to varying degrees of noise added to the pattern data used in the previous section. We use the parameters that optimized the results for the patterned data ($k = 6, \theta = 30\%$). Results are given in Table 6.15 and Figure 6.8.

There are two aspects to how noise can interfere with the data mining process. First, if the data mining process cannot differentiate pattern items from the noise items, precision will decrease. In **ApproxMAP**, pattern items are identified as those items that occur regularly in a certain location after alignment. The probability of a random item showing up in a certain position frequently enough to be aligned is very low. Thus, even with large amounts of noise present in the data, these random noise are easily identified and ignored as noise.

In fact, Figure 6.8(b) shows that precision actually increases when there is more noise in the data. This is because with noise in the data, **ApproxMAP** is more likely to miss the weak signatures that it identified as patterns without noise. For recoverability, that means that some weak signatures that are true patterns will be missed resulting in some reduction in recoverability. On the other hand, for precision, that means that the weak signatures that were false positives will now be accurately identified as extraneous items resulting in higher

Figure 6.8: Effects of noise ($k = 6, \theta = 30\%$)

precision. Recently, researchers have looked into intentionally adding noise to data for such reasons [] [FROM KUM: **add reference**].

Second, noise can interfere with detecting the proper pattern items. This will reduce recoverability. As discussed in the previous paragraph, noise will cause **ApproxMAP** to miss some of the weak signatures and in turn reduce recoverability slightly. However, when the base pattern appears fairly frequent in the database **ApproxMAP** can still detect most of the base pattern even in the presence of noise.

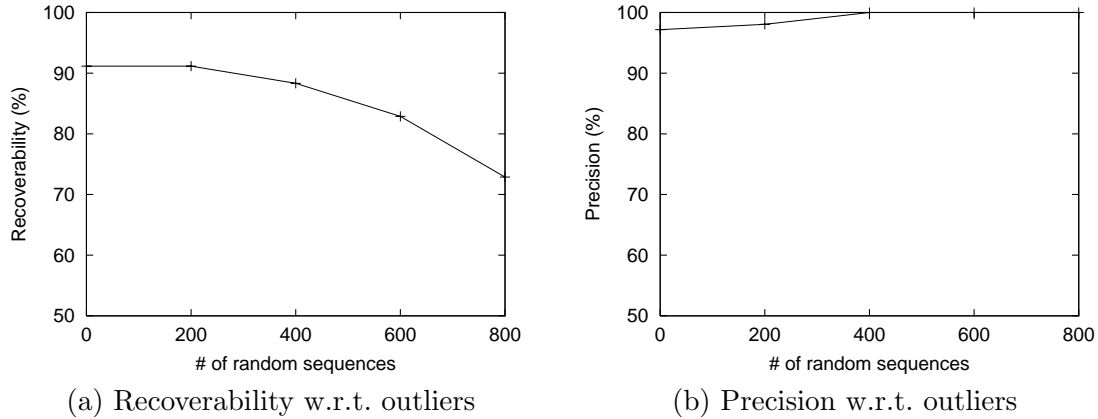
To understand why, let us consider the behavior of **ApproxMAP** in the presence of noise. When there are noise in a data sequence, **ApproxMAP** will simply align based on the remaining pattern items. Then once the alignment is done **ApproxMAP** reports on the items that occur regularly in each position. Note that the items that are randomly changed are different in each sequence. Thus, a particular pattern item missing from some of the data sequences, is most likely still captured by **ApproxMAP** because there will be other data sequence with that particular pattern item in the proper position if the base pattern signature is strong enough.

Furthermore, **ApproxMAP** aligns the sequences by starting with the most similar sequences and working out to the least similar. The weighted sequence built from the core of the sequences in the cluster forms a center of mass. That is, the items in the base pattern start to emerge in certain positions. Then the sequences with more noise can easily attach its pattern items to the strong underlying pattern emerging in the weighted sequence. In essence, **ApproxMAP** works only with those items that align with other sequences and simply ignores those that cannot be aligned well. This makes **ApproxMAP** very robust to noise in the data.

Table 6.15 and Figure 6.8(a) show that the results are fairly good with up to 40% of noise in the data. Despite the presence of noise, it is still able to detect a considerable number of the base patterns (i.e. recoverability is 88.21% when corruption factor is 40%) with no extraneous items (precision is 100%) or spurious patterns. At 50% of noise (that is 50% of

Table 6.16: Effect of outliers ($k = 6, \theta = 30\%$)

$N_{outlier}$	N_{PatSeq}	N_{PatI}	N_{ExtraI}	\mathcal{R}	\mathcal{P}	N_{total}	N_{spur}	N_{redun}	$\ C_{null}\ $	$\ C_{one_itemset}\ $	$\ C\ $
0	1000	103	3	91.16%	97.17%	8	0	1	1	0	9
200	1000	100	4	91.16%	98.04%	8	0	1	2	0	10
400	1000	97	6	88.33%	100%	8	0	1	3	0	11
600	1000	92	10	82.87%	100%	8	0	1	4	0	12
800	1000	81	11	72.88%	100%	7	0	1	10	1	18

Figure 6.9: Effects of outliers ($k = 6, \theta = 30\%$)

the items in the data were randomly changed to some other item or deleted), recoverability starts to degenerate (recoverability=64.03%).

Clearly, the results show that **ApproxMAP** is robust to noise in the data. **ApproxMAP** is very robust to noise because it looks for long sequential patterns in the data and simply ignores all else.

6.2.4 Experiments 2.4: Robustness with respect to outliers

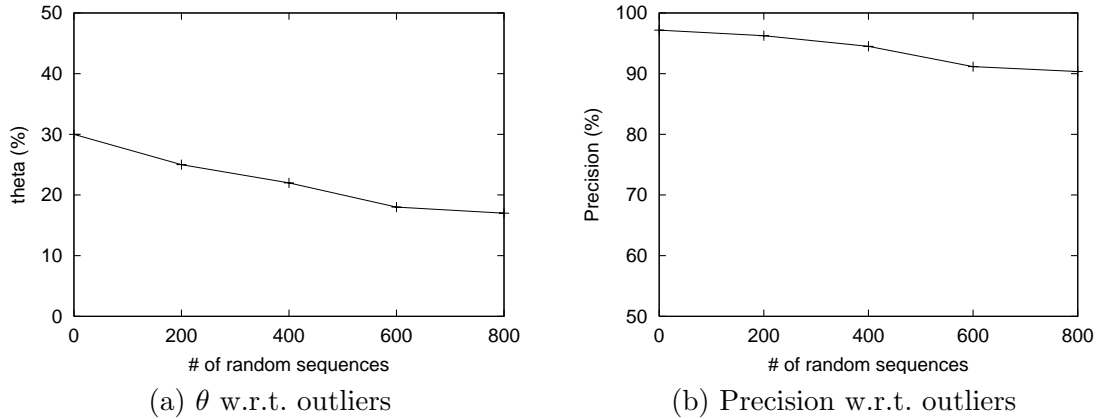
This experiment is designed to test the effect of outliers in the data. To do so, we added varying number of outliers (random sequences) to patterned data used in experiment 2.2. The main effect of the outliers are the weakening of the patterns as a percentage of the database.

Again we start with the parameters that optimized the results for the patterned data ($k = 6, \theta = 30\%$). When k is maintained at 6, the added random sequences had no effect on the clusters formed or how the patterned sequences were aligned because the nearest neighbor list does not change much for the patterned data. Each cluster just picks up various amounts of the outliers which are aligned after all the patterned sequences are aligned. In effect, the outliers are ignored when it cannot be aligned with the patterned sequences in the cluster. The rest of the outliers formed small separate clusters that generated no patterns, as in experiment 2.1 on random data.

Nonetheless, the consensus sequences were shorter because the random sequences increased

Table 6.17: Effect of outliers ($k = 6$)

$N_{outlier}$	N_{PatSeq}	θ	N_{PatI}	N_{ExtraI}	\mathcal{R}	\mathcal{P}	N_{total}	N_{spur}	N_{redun}	$\ C_{null}\ $	$\ C_{one_itemset}\ $	$\ C\ $
0	1000	30%	103	3	91.16%	97.17%	8	0	1	1	0	9
200	1000	25%	103	4	91.16%	96.26%	8	0	1	2	0	10
400	1000	22%	103	6	91.16%	94.50%	8	0	1	3	0	11
600	1000	18%	103	10	91.16%	91.15%	8	0	1	4	0	12
800	1000	17%	103	11	91.16%	90.35%	8	0	1	10	0	18

Figure 6.10: Effects of outliers ($k = 6$)

the cluster size (Table 6.16 and Figure 6.9). This in turn weakened the signature in the cluster resulting in reduced recoverability (Figure 6.9(a)) when the cutoff θ was kept at the same level (30%) as the pattern data. Similar to what we saw with noise in the data, precision slightly increases with more outliers (Figure 6.9(b)).

However, as seen in Table 6.17, we can easily find the longer underlying patterns by adjusting θ to compensate for the outliers in the data. That is θ can be lowered to pick up more patterned items. The tradeoff would be that more extraneous items could be picked up as well. We assessed the tradeoff by finding out how many extraneous items would be picked up by **ApproxMAP** in order to detect the same number of patterned items as when there were no outliers. In Table 6.17 the largest θ for which the exact same patterned items can be found are reported on each dataset along with the additional extraneous items picked up. We include the number of null and one-itemset clusters as well.

We can see in Figure 6.10, that with only a small drop in θ we can recover all of the base patterns detected without any outliers with only a small decrease in precision for all dataset. In general, the more outliers, the more extraneous items are found for the threshold that gives the same patterned items.

Again, clearly **ApproxMAP** is robust to outliers. Even with over 40% outliers in the data ($\frac{800}{1000} \cdot 100\% = 44.4\%$) precision is good at 90.35% when θ is set to give the same recoverability as the patterned data (91.16%). This is inline with the results of experiment 2.1 on random data. Random data has very little affect on **ApproxMAP**. It does not introduce spurious

Table 6.18: Parameters for the data generator for Experiment 3

Notation	Meaning	Default value
$\ I\ $	# of items	1,000
$\ \Lambda\ $	# of potentially frequent itemsets	5,000
N_{seq}	# of data sequences	10,000
N_{pat}	# of base pattern sequences	100
L_{seq}	Avg. # of itemsets per data sequence	20
L_{pat}	Avg. # of itemsets per base pattern	$14 = 0.7 \cdot L_{seq}$
I_{seq}	Avg. # of items per itemset in the database	2.5
I_{pat}	Avg. # of items per itemset in base patterns	$2 = 0.8 \cdot I_{seq}$

Table 6.19: Parameters for ApproxMAP for Experiment 3

k	# of neighbor sequences	5
θ	the strength cutoff for pattern consensus sequences	50%
$min_DB_strength$	the optional parameter for pattern consensus sequences	10 sequences

patterns and it does not hinder ApproxMAP in finding the true base patterns. The only real effect is the increased data size which in turn weakens the pattern as a proportion of the data. This can easily be compensated for by adjusting θ .

6.3 Database Parameters And Scalability

Now that we have a good understanding of ApproxMAP and its effectiveness, we go on to study its performance on different types of patterned data. Here we examine the effects of various parameters of the IBM patterned database on the results. We use much bigger datasets in these experiments. The default configuration of the data sets used is given in Table 6.18. The expected frequencies of the 100 base patterns embedded in the dataset range from 7.63% to 0.01%. The full distribution of $E(F_B)$ is given in Figure 5.1(b). See section 5.3 for a detailed discussion on the properties of this synthetic database. We used all default parameters for ApproxMAP which are restated in Table 6.19.

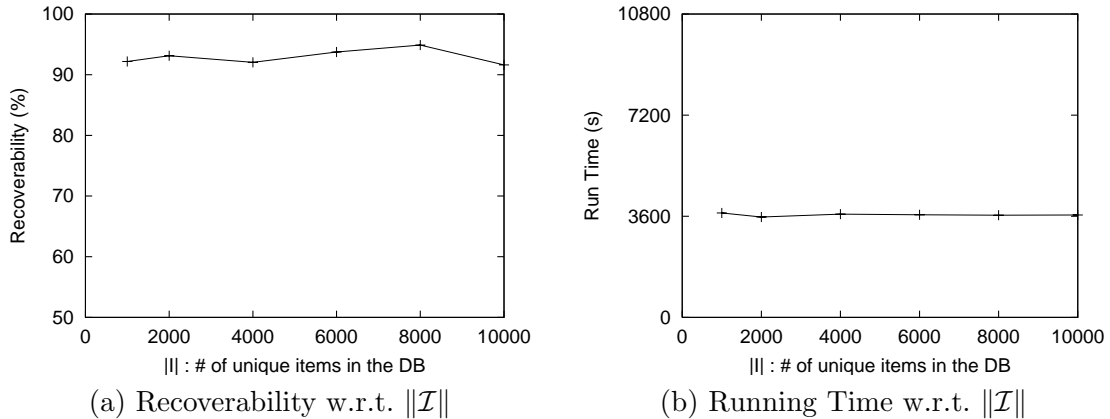
We study the results on four factors, the total number of unique items in the data set $\|\mathcal{I}\|$ and the data set size in terms of (1) number of sequences, N_{seq} , (2) the average number of itemsets in a sequence, L_{seq} , and (3) the average number of items per itemset, I_{seq} . The following analysis strongly indicates that ApproxMAP is effective and scalable in mining large databases with long sequences.

6.3.1 $\|\mathcal{I}\|$: Number of unique items in the database

We first studied the effects of the number of items in the set \mathcal{I} , $\|\mathcal{I}\|$. A smaller value of $\|\mathcal{I}\|$ results in a denser database (i.e., patterns occur with higher frequencies) because the

Table 6.20: Results for $\|\mathcal{I}\|$

$\ \mathcal{I}\ $	Recoverability	Precision	N_{spur}	N_{redun}	N_{total}	Running Time(s)
1000	92.17%	100.00%	0	18	94	3718
2000	93.13%	100.00%	0	15	91	3572
4000	92.03%	100.00%	0	22	96	3676
6000	93.74%	100.00%	0	15	94	3652
8000	94.89%	100.00%	0	16	92	3639
10000	91.62%	100.00%	0	15	85	3644

Figure 6.11: Effects of $\|\mathcal{I}\|$ 

items come from a smaller set of literals. In multiple alignment, the positions of the items have the strongest effect on the results. Although same items may occur often (and belong to many different base patterns), if the item occurs in different locations in relation to other items in the sequence the items can be easily identified as coming from different base patterns. Thus, even when the density of the database changes, the alignment statistics does not change substantially in our experiments. The full results are given in Table 6.20 and Figure 6.11. We observe that there is no noticeable difference in the results of **ApproxMAP** in terms of the five evaluation measures as well as the running time. In this experiment, for a wide range of $\|\mathcal{I}\|$ between 1,000 to 10,000,

1. recoverability is consistently over 90%,
2. there is no extraneous items (precision=100%) or spurious patterns,
3. there is a manageable number of redundant patterns ($15 \geq N_{redun} \geq 22$),
4. and running time is constant at about 1 hour.

6.3.2 N_{seq} : Number of sequences in the database

In this section, we test the effect of data set size in terms of number of sequences in the data set. The results are shown in Table 6.21 and Figure 6.12. As the data set size increases with respect to N_{seq}

Table 6.21: Results for N_{seq}

N_{seq}	Recoverability	Precision	N_{spur}	N_{redun}	N_{max}	N_{total}	Running Time(s)
10000	92.17%	100.00%	0	18	76	94	3718
20000	95.81%	100.00%	0	39	87	126	12279
40000	98.39%	99.39%	0	54	93	147	45929
60000	99.29%	96.44%	0	107	95	202	98887
80000	98.76%	95.64%	0	134	95	229	159394
100000	98.90%	93.45%	0	179	97	276	232249

1. recoverability increases slightly,
2. precision decreases slightly,
3. the number of max patterns increase,
4. the number of redundant patterns increase,
5. there is no spurious patterns,
6. and running time is quadratic (very modest slope) with respect to N_{seq} .

Let us look at each point in more detail. It is interesting to note that the recoverability increases as the data set size goes up, as shown in 6.12(a). It can be explained as follows. With multiple alignment, the more the sequences in the data set, the easier the recovery of the base patterns. In large data sets, there are more sequences approximating the patterns. For example, if there are only 1000 sequences, a base pattern that occurs in 1% of the sequences will only have 10 sequences approximately similar to it. However, if there are 100,000 sequences, then there would be 1,000 sequences similar to the base pattern. It would be much easier for **ApproxMAP** to detect the general trend from 1,000 sequences than from 10 sequences.

This is the same reason why overall there are more patterns returned, N_{total} , as N_{seq} increases. When there are more sequences, **ApproxMAP** is able to detect more of the less frequent patterns. Some of the extra sequences detected are categorized as redundant patterns, hence the increase in N_{redun} , and others are max patterns, N_{max} . Obviously, the increase in recoverability is directly related to the increase in max patterns.

Remember redundant patterns returned are almost never the exact same patterns being returned. As shown in the small example in Table 6.14, redundant patterns are the consensus sequences that come from the same base patterns ($PatConSeq_2$ and $PatConSeq_3$). Many times redundant patterns hold additional information about possible variations to the max pattern. For example, $PatConSeq_2$ and $PatConSeq_3$ in the small example suggests that there might be many sequences in the dataset that are missing the later part of the longer $PatConSeq_2$ ². As there are more sequences in the dataset, **ApproxMAP** is more likely to pick up these variations, which results in increased redundant patterns.

²Such hypothesis can be verified using more efficient pattern search methods.

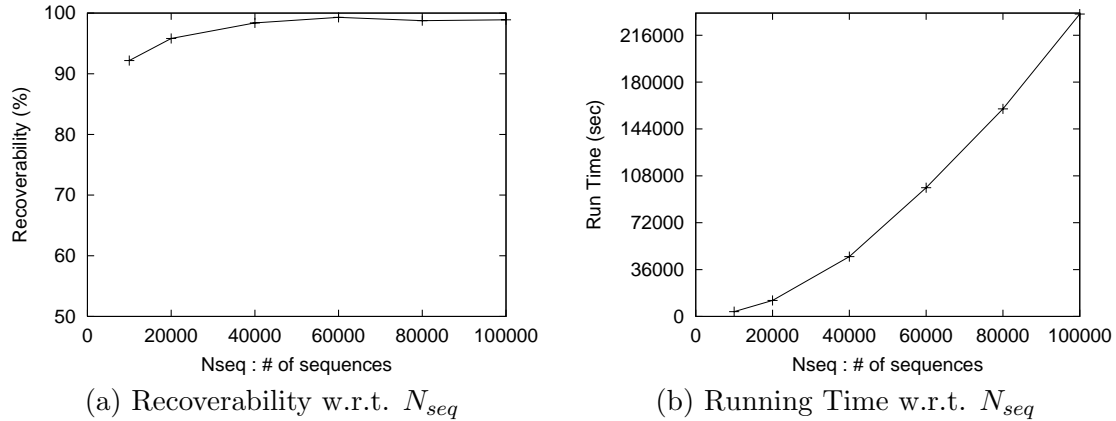
Figure 6.12: Effects of N_{seq} 

Table 6.22: A new underlying trend emerging from 2 base patterns

ID	len	Patterns							
PatC	16	<u>(22,43)</u>	<u>(2,22,28,73)</u>	<u>(96)</u>	<u>(95)</u>	<u>(2,74)</u>	<u>(5,24,65,93)</u>	<u>(2,6)</u>	
BP ₁	14	(43)	(2,28,73)	(96)	(95)	(2,74)	(5)	(2)	(24, 63) (20) (93)
BP ₂	14	(22)	(22)	(58)	(2, 16, 24, 63)	(24,65,93)	(6)		(11, 15, 74)

Another reason for the increase in redundant patterns as N_{seq} increases is due to the limitations in our evaluation method discussed in section 5.4. That is, the current evaluation method maps each result pattern to only one base pattern. However, this is not always accurate. Essentially, when N_{seq}/N_{pat} is large in the IBM data generator, a new underlying pattern can emerge as two base patterns combine in a similar way frequently enough. This new mixed pattern will occur regularly in the database and a good algorithm should detect it.

To demonstrate how ApproxMAP works in such situations, we look in detail at an example from one of our experiments³. In Table 6.22, *PatC* is the pattern consensus sequence returned for a cluster. The underlined items are the 6 extraneous items as determined by the evaluation method. Clearly, *PatC* originates from base pattern *BP₁* since the two share 10 items. The color of the items in *PatC* indicates the strength of the items while the dark items for *BP₁* are shared items and the light items are missed items. However, it is also clear that the remaining 6 items in *PatC* (the underlined items) originated from *BP₂*. This cluster seems to suggest that there is a group of sequences that combine the two base sequences, *BP₁* and *BP₂*, in a similar manner to create a new underlying pattern similar to *PatC*.

³Instead of using an example from one of the experiments done in this section we use a smaller experiment because it is much easier to see. The example had the following parameters for the data generator: $N_{seq} = 5000$, $N_{pat} = 10$, $L_{seq} = 10$, $L_{pat} = 7$, $I_{seq} = 2.5$, and $I_{pat} = 2$.

Table 6.23: The full aligned cluster for example given in Table 6.22

BP ₁	{43}	{2,28,73}	{96}		{95}	{2,74}	{5}	{2}	{24,63}	{20}	{93}
BP ₂	{22}	{22}	{58}		{2,16,24,63}		{24,65,93}	{6}	{11,15,74}		
Wgt	{15:1,16:1, 17:1,22:16, 31:2,43:15, 63:4,66:1, 76:2}:16	{2:16,15:1, 16:1,17:1, 22:13,28:16, 66:1,73:16, 88:2}:16	{2:3,15:1, 16:1,22:8, 58:8,63:1, 66:1,96:16} :1	{58:1, 63:1, 95:1, 96:1}	{2:9,16:9,22:2,24:9, 30:1,58:7,63:9,66:1, 74:2,95:16}:16	{2:16,5:1,16:2, 22:1,24:6,31:1, 63:1,65:5,74:16, 76:1,93:5}:16	{5:11,6:2,16:1, 24:12,31:1,58:1, 65:11,66:1,76:1, 93:11}:15	{2:14,6:15,11:2, 15:2,22:2,50:6, 58:2,62:2,66:4, 74:2,96:1}:16	{2:1,6:1,11:3, 15:3,16:1,24:3, 50:1,62:1,63:3, 74:3}:4	{20:1, 22:1, 50:1, 66:1}	
PatC	{22,43}	{2,22,28,73}	{96}		{95}	{2,74}	{5,24,65,93}	{2,6}			
VarC	{22,43}	{2,22,28,73}	{22,58,96}		{2,16,24,58,63,95}	{2,74}	{5,24,65,93}	{2,6}			
763	{22,43}	{2,22,28,73}	{96}		{58,95}	{2,74}	{24,65,93}	{2,6}			
762	{22,43}	{2,22,28,73}	{58,96}		{2,16,24,63,95}	{2,74}	{5,24,65,93}	{2,6}			
764	{22,43}	{2,22,28,73}	{96}		{58,95}	{2,74}	{5,24,65,93}	{2,6}			
767	{22,43}	{2,28,73}	{22,96}		{58,95}	{2,74}	{5,24,65,93}	{2,6}			
761	{22,43}	{2,22,28,73}	{22,58,96}		{2,16,24,63,95}	{2,24,65,74,93}	{5,6,24,65,93}	{2,6,11,15,74}			
775	{22,43,63}	{2,22,28,73}	{2,22,58,96}		{2,16,24,63,95}	{2,74}	{5,24,65,93}	{2,6,50}			
718	{22}	{2,22,28,73}	{58,96}		{95}	{2,74}	{24,65,93}	{6}			
766	{22,43}	{2,22,28,73,88}	{96}		{58,95}	{2,22,74}	{24,65,93}	{2,6,58}			
1133	{22,43}	{2,22,28,73}	{22,96}		{58,95}	{2,16,24,63,74}	{24,65,93}	{2,6}	{6,11,15,24, 63,74}		
765	{22,43}	{2,22,28,73,88}	{22,58,63,96}		{2,16,24,58,63,95}	{2,24,65,74,93}	{5,24,58,65,93}	{2,6,22,50,66,96}			
743	{22,31,43,76}	{2,22,28,73}	{58,96}		{58,66,95}	{2,74}	{5,16}	{2,6,50,62,66}			
590	{15,16,17,22, 43,63,66}	{2,22,28,73}	{2,22,96}		{2,16,22,24,63,74,95}	{2,31,74,76}	{5,66}	{2,6,50,58}			
776	{22,43,63}	{2,28,73}	{2,22,96}		{2,16,24,63,95}	{2,24,65,74,93}	{5,6}	{2,11,15,74}			
742	{22,31,43,76}	{2,22,28,73}	{58,66,96}		{2,16,22,24,30,63,95}	{2,16,24,65,74,93}		{6,50,62,66}			
1909	{22,43}	{2,28,73}	{22,96}	{58,63, 95,96}	{2,16,24,63,74,95}	{2,5,74}	{5,24,65,93}	{2,6}	{11,15,74}	{20,22, 50,66}	
1145	{22,43,63}	{2,15,16,17,22, 28,66,73}	{15,16,58,96}		{2,16,24,63,95}	{2,24,65,74,93}	{5,24,31,76}	{2,6,22,50,66}	{24,50,62,63}		

Table 6.24: Results for N_{seq}

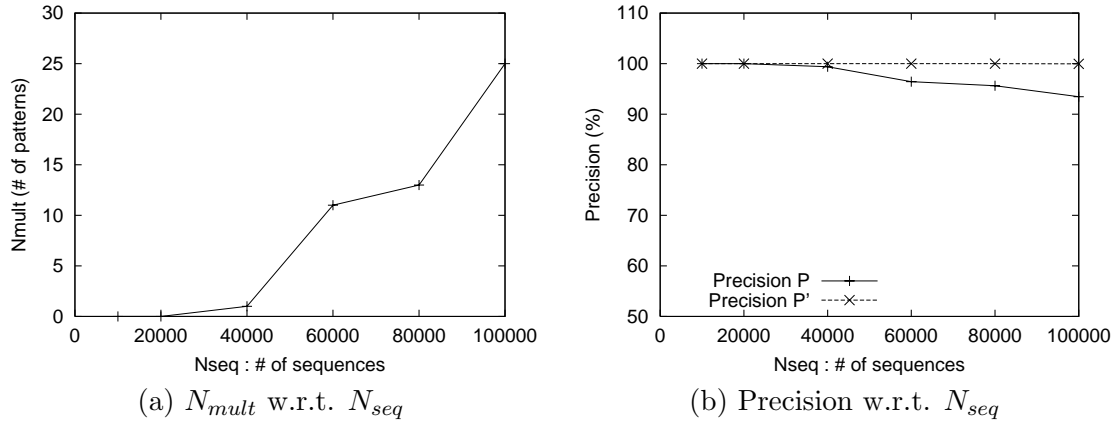
N_{seq}	Precision \mathcal{P}	N_{item}	N_{patI}	N_{extraI}	N_{patI2}	$N_{extraI2}$	Precision \mathcal{P}'	N_{mult}
10000	100%	2245	2245	0	0	0	100%	0
20000	100%	3029	3029	0	0	0	100%	0
40000	99.39%	3590	3568	22	22	0	100%	1
60000	96.44%	5086	4905	181	180	1	100%	11
80000	95.64%	5662	5415	247	247	0	100%	13
100000	93.45%	7374	6891	483	479	4	99.95%	25

We investigate this cluster further in Table 6.23. Table 6.23 gives the full aligned cluster (size=16 sequences). $WgtSeq$ is the weighted sequence for the cluster. $PatC$ is the pattern consensus sequence with $\theta = 50\%$, and $VarC$ is the variation consensus sequence with $\delta = 40\%$. δ was set higher than the default because the cluster size was very small (16 sequences). Note that since $\theta = 50\% = 8 \text{ seq} < \text{min_DB_strength} = 10 \text{ seq}$, the cutoff for pattern consensus sequence is 10 sequences. The cutoff for the variation consensus sequence is $\delta = 40\% = 7$ sequences. The aligned sequences clearly show that the underlying pattern in the data is a combination of the two base patterns BP_1 and BP_2 . We see that all 16 items in the consensus sequence are approximately present in all sequences. There are no real extraneous items.

Furthermore, the five items in the middle of BP_2 , $\langle(58)(2, 16, 24, 63)\rangle$, are present in the data sequence with slightly less frequency (between 10 and 7 sequences). These items are accurately captured in the variation consensus sequence. In comparison, the seven items at the end of the two base patterns, $\langle(24, 63)(20)(93)\rangle$ for BP_1 and $\langle(11, 15, 74)\rangle$ for BP_2 , barely exist in the data. All are present in less than 5 sequences. Thus, these items are not included in the consensus pattern because it does not exist frequently in the data. It is clear from looking at the aligned sequences that ApproxMAP has accurately mapped the 16 sequences to the combined pattern underlying the cluster.

Detecting such patterns could be quite useful in real applications for detecting interesting unknown patterns, which arise by systematically combining two known patterns. For example, we might know the separate buying patterns of those that smoke and those that drink. However if there are any systematic interactions between the two groups that were not known ApproxMAP could detect it.

In Table 6.24, we show the number of extraneous items that could be built into a second base pattern in column N_{patI2} and the remaining true extraneous items in $N_{extraI2}$. All but 1 of 181 extraneous items, when $N_{seq} = 60K$, could be built into a second base pattern. For $N_{seq} = 100K$, all but 4 of 483 extraneous items came from a second base pattern. In all other experiments, all the extraneous items came from a second base pattern. Column Precision \mathcal{P}' gives the modified precision which exclude the extraneous items that map back to a second base pattern. Column N_{mult} is the number of consensus patterns that mapped back to 2

Figure 6.13: Effects of N_{seq} **Table 6.25: Results for L_{seq}**

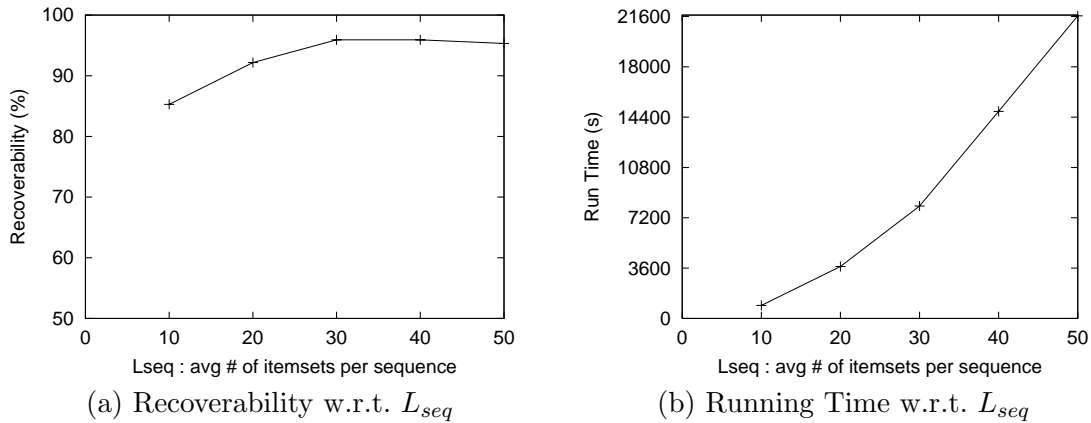
L_{seq}	Recoverability	Precision	N_{spur}	N_{redun}	N_{total}	Running Time(s)
10	85.29%	100.00%	0	26	92	934
20	92.17%	100.00%	0	18	94	3718
30	95.92%	100.00%	0	16	94	8041
40	95.93%	100.00%	0	11	93	14810
50	95.32%	100.00%	0	16	93	21643

base patterns for each experiment. Not surprisingly, as N_{seq} increases (which will increase N_{seq}/N_{pat}), more interacting patterns (patterns that are a systematic mix of multiple base patterns) are found (Figure 6.13(a)).

In terms of the current evaluation method, there are two consequences of detecting these new patterns. First, the detected pattern will most likely be categorized as a redundant pattern. This is because, statistically it is very likely that there will be another cluster which represents the primary base pattern well. Thus, the small cluster giving the new emerging pattern is counted as a redundant pattern.

Second, the items that do not map back to the primary base pattern will be counted as extraneous items. Consequently, as N_{seq} increase, and more interacting patterns are found, precision decrease. However, when the second base pattern is properly taken into account we see that precision, \mathcal{P}' , is stable as N_{seq} increases (Figure 6.13(b)).

In terms of running time, we observe that **ApproxMAP** is scalable with respect to the data set size, as shown in Figure 6.12(b). Although asymptotically **ApproxMAP** is quadratic with respect to N_{seq} , we see that practically it is close to linear. Furthermore, when the dataset size is really big ($N_{seq} \geq 40K$), **ApproxMAP** can use the sample based iterative clustering method discussed in section 4.6.2.

Figure 6.14: Effects of L_{seq} Table 6.26: Results for I_{seq}

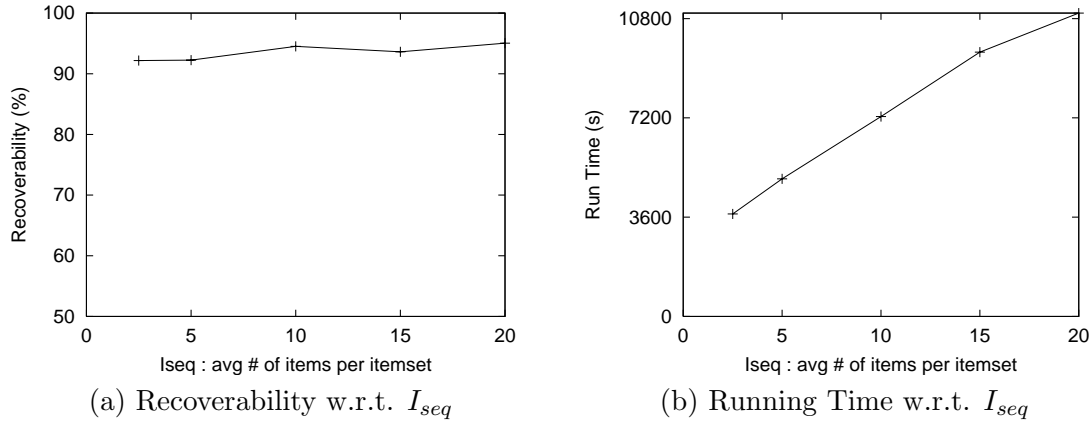
I_{seq}	Recoverability	Precision	N_{spur}	N_{redun}	N_{total}	Running Time(s)
2.5	92.17%	100.00%	0	18	94	3718
5	92.25%	100.00%	0	21	95	4990
10	94.52%	100.00%	0	21	98	7248
15	93.61%	100.00%	0	30	104	9583
20	95.05%	100.00%	0	26	105	11000

6.3.3 $L_{seq} \cdot I_{seq}$: Length of sequences in the database

We observe similar results with respect to the length of the sequences in the database. As the length increases, the results improve. There are two parameters that control the length (total number of items) of a sequence. L_{seq} is the average number of itemsets in a sequence and I_{seq} is the average number of items per itemset. Therefore, both change the total number of items of the data sequences as well as the embedded base patterns. As either parameter increases, the additional items provide more clues for proper alignment leading to better results.

More itemsets per sequence (larger L_{seq}), has a direct effect on improving alignment due to more positional information during alignment. That is, as the average number of itemsets in a sequence increase, recoverability tends to increase (Figure 6.14(a)). As the number of elements in the sequence (itemsets) increase the pattern signature becomes longer. That means there are more clues as to what is the proper alignment. This in turn makes it easier to capture the signature through alignment. The result are given in Table 6.25 and Figure 6.14.

On the other hand, more items per itemset (larger I_{seq}) has an indirect effect of improving alignment by providing more evidence on the location of a particular position. For example, if $I_{seq} = 2.5$, that means that on average there is only 2.5 items in any position to indicate its

Figure 6.15: Effects of I_{seq} 

proper position. If one or two items in a particular itemset for a sequence is missing due to variations, there is not much more information left to use to identify the proper alignment. In comparison, if $I_{seq} = 10$, that means that on average there are 10 items in any position. So, even if one or two items are missing due to slight variations in the data, there are still a good number of items that can be used to locate the proper position in the alignment. The indirect effect of I_{seq} is not as pronounced as L_{seq} . The results are given in Table 6.26 and Figure 6.15.

Basically, the longer the pattern the more likely it is to detect the patterns. Thus, both L_{seq} and I_{seq} have similar affects, but L_{seq} has a stronger affect. In summary, for a wide range of L_{seq} between 10 to 50:

1. Recoverability tends to be better as the sequences get longer and levels off at around 96% at $L_{seq} = 30$.
2. There is no extraneous items (precision=100%) or spurious patterns.
3. There is a manageable number of redundant patterns ($16 \geq N_{redun} \geq 26$).
4. And the running time is close to linear with respect to L_{seq} with the optimization discussed in section 4.6.1 (Figure 6.14(b)).

The experiment of $I_{seq} = 2.5..20$ gave similar results as follows:

1. Recoverability has a slight upward tendency (Figure 6.15(a)).
2. There are no extraneous items (precision=100%) or spurious patterns.
3. And there is a manageable number of redundant patterns ($18 \geq N_{redun} \geq 30$).
4. However, unlike L_{seq} , the running time is linear with respect to the average number of items per itemset, I_{seq} . The time complexity analysis in section 4.5 show that ApproxMAP should be linear with respect to I_{seq} . Our experiment confirmed the relationship as shown in Figure 6.15(b).