An Efficient Method for Discovering Motifs in Large Time Series

Cao Duy Truong and Duong Tuan Anh

Faculty of Computer Science and Engineering,
Ho Chi Minh City University of Technology
caoduytruong@hcmunre.edu.vn, dtanh@cse.hcmut.edu.vn

Abstract. Time series motif is a previously unknown pattern appearing frequently in a time series. An efficient motif discovery algorithm for time series would be useful as a tool for summarizing massive time series databases as well as many other advanced time series data mining tasks. In this paper, we propose a new efficient algorithm, called EP-BIRCH, for finding motifs in large time series datasets. This algorithm is more efficient than MK algorithm and stable to the changes of input parameters and these parameters are easy to be determined through experiments. The instances of a discovered motif may be of different lengths and user does not have to predefine the length of the motif.

Keywords: time series motif, motif discovery algorithm, clustering algorithm, extreme points.

1 Introduction

A time series is a sequence of real numbers measured at equal intervals. Time series data arise in so many applications of various areas ranging from science, engineering, business, finance, economic, medicine to government. Nowadays, time series datasets in several applications becomes very large, with the scale of multi-terabytes and data mining task in time series data of such scale becomes very challenging.

Time series motifs are frequently occurring but previously unknown subsequences of a longer time series, which are very similar to each other. Since their formalization in 2002 by Lin et al. [4], several time series motif discovery algorithms have been proposed ([1], [4], [5], [6], [7], [12]). With most of these algorithms, user has to determine in advance the length of the motif and the distance threshold (range) for subsequence matching, which are the two parameters in most of the motif discovery algorithms. There have been a few algorithms proposed for finding time series motif with different lengths or variable lengths ([3], [9], [10]). However so far, to the best of our knowledge, there has been no time series motif discovery algorithm that can determine automatically the suitable length of the motif in a time series.

In this paper, we propose a new efficient algorithm for detecting time series motif which uses significant extreme points to determine motif candidates and then cluster motif candidates to find the most significant motif by using BIRCH algorithm. This

A. Selamat et al. (Eds.): ACIIDS 2013, Part I, LNAI 7802, pp. 135–145, 2013.

algorithm works directly in the original time series without any transformation for dimensionality reduction. The experiments on the real world datasets demonstrate that our proposed method outperforms the MK algorithm [6]. Besides, the proposed algorithm has three other major advantages. First, it can perform effectively with large time series datasets. Second, it is not sensitive to input parameters and these parameters can be determined easily. Third, the instances of a discovered motif may be of different lengths and user does not have to predefine the length of the motif.

2 Background

2.1 Time Series Motif and Bruce-Force Algorithm for Finding Motifs

Definition 1. *Time Series:* A time series $T = t_1,\ldots,t_N$ is an ordered set of N real-values measured at equal intervals.

Definition 2. *Similarity distance:* $D(s_1, s_2)$ is a positive value used to measure differences between two time series s_1, and s_2, relies on measure methods. If $D(s_1, s_2) < r$, where r is a real number (called *range*), then s_1 is similar to s_2 .

Definition 3. *Subsequence*: Given a time series T of length N, a subsequence C of T is a sampling of length $n < N$ of contiguous positions from T, that is, $C = t_p,\ldots,t_{p+n-1}$ for $1 \le p \le m - n + 1$.

Definition 4. *Time series motif:* Given a time series T, a subsequence C is called the most significant motif (or 1-motif) of T, if it has the highest count of the subsequences that are similar to it. All the subsequences that are similar to the motif are called *instances* of the motif.

Lin et al. ([4]) also introduced the brute-force algorithm to find the most significant motif (see Fig. 1). The input parameters of this algorithm are the length of the motif (n) and the distance threshold (r). Notice that the brute-force algorithm enumerates the non-trivial matches between subsequences. Given a time series T containing a subsequence C beginning at position p and a matching subsequence M beginning at q, we say that M is a *trivial match* to C if either $p = q$ or there does not exists a subsequence M' beginning at q' such that $D(C, M') > r$ and either $q<q'<p$ or $p<q'<q$. In the brute-force algorithm, we exclude the trivial matches so that only non-trivial matches are counted for detecting 1-motif in a time series.

This brute-force algorithm work directly on raw time series and requires $O(m^2)$ calls to the distance function (m is the length of the time series).

Algorithm Find-1-Motif-Brute-Force(*T, n, r*)
best_motif_count_so_far = 0
best_motif_location_so_far = null;
for $i = 1$ **to** length(T) – n + 1
  count = 0; pointers = null;
  for $j = 1$ **to** length(T) – n + 1

```
    if Non_Trivial_Match (C[i: i + n – 1], C[j: j + n – 1], r ) then
        count = count + 1;
        pointers = append (pointers, j);
    end
  end
  if count > best_motif_count_so_far then
    best_motif_count_so_far = count;
    best_motif_location_so_far = i;  motif_matches = pointers;
  end
end
```

Fig. 1. The outline of brute-force algorithm for 1-motif discovery

2.2 Algorithms for Finding Motifs

Since the definition of time series motif was given in 2002 [4], several algorithms have been proposed to tackle the problem of time series motif discovery. The first algorithm that can find motifs in linear time is Random Projection, developed by Chiu et al. in 2003 [1]. It is an iterative approach and uses as base structure a collision matrix whose rows and columns are the SAX representation of each time series subsequence.

Mueen et al. in 2009 [6] proposed the first exact motif discovery algorithm, called MK algorithm, that works directly on raw time series data. One of the major disadvantages of the Random Projection algorithm and the MK algorithm is that they still execute very slowly with large time series data.

In this work, we improve the method for time series motif discovery proposed by Gruber et al. 2006 [2]. This method is based on the concepts of significant extreme points that was proposed by Pratt and Fink, 2002 [8]. The algorithm proposed by Gruber et al. for finding time series motifs consists of three steps: extracting significant extreme points, determining motif candidates from the extracted significant extreme points and clustering the motif candidates to determine the 1-motif through the cluster with the largest numbers of candidates. For convenience, in this paper, we call the algorithm proposed by Gruber et al. EP-C (Extreme Points and Clustering). When Gruber et al. proposed the EP-C algorithm, they apply it in signature verification and did not compare it to any previous time series motif discovery algorithms. Through our experiments done by Tin, 2012 [11], we found out that the EP-C is much more effective than Random Projection in terms of time efficiency and motif accuracy. But EP-C still needs some improvements.

2.3 Finding Time Series Motifs Using the MK Algorithm

Mueen et al. in 2009 [6] proposed the first exact motif discovery algorithm, called MK algorithm, that works directly on raw time series data. This algorithm uses the "nearest neighbor" definition of motif as follows. Time series motifs are pairs of subsequences which are very similar to each other.

Based on MK algorithm, we can modify it so that it can detect 1-motif in time series according the first formalized definition given by Lin et al. [4]. The modification can be done simply as follows: for each *i*-th subsequence of a longer time series, we use a linked list to store all the subsequences that match with the *i*-th subsequence. Later, the linked list with the largest number of matching subsequences will be the linked list associated with the 1-motif of the time series.

In the modified MK algorithm, we can also apply the three improvement techniques proposed by Mueen et al., 2009 ([6]). The three improvement techniques are (i) exploiting the symmetry of Euclidean distance, (ii) exploiting triangular inequality and reference point, and (iii) applying early abandoning.

Thank to the three techniques, the modified MK algorithm is up to three orders of magnitude faster than brute-force algorithm. More details about the three improvement techniques, interested reader can refer to [6]. In this work, we will use the modified MK algorithm as the baseline algorithm to compare with our proposed algorithm for finding time series motif.

2.4 Finding Significant Extreme Points

To extract a temporally ordered sequence of motif candidates, significant extreme points of a time series have to be found. The definition of significant extreme points, given by Pratt and Fink, 2002 [8] is as follows.

Definition 5. *Significant Extreme Points:* A univariate time series $T = t_1,\ldots,t_N$ has a *significant minimum* at position m with $1 < m < N$, if $(t_i, \ldots, t_j)$ with $1 \le i < j \le N$ in T exists, such that t_m is the minimum of all points of this subsequence and $t_i \ge R \times t_m$, $t_j \ge R \times t_m$ with user-defined $R \ge 1$.
Similarly, a *significant maximum* is existent at position m with $1 < m < N$, if a subsequence $(t_i, \ldots, t_j)$ with $1 \le i < j \le N$ in T exists, such that t_m is the maximum of all points of this subsequence and $t_i \le R \times t_m$, $t_j \le R \times t_m$ with user-defined $R \ge 1$.

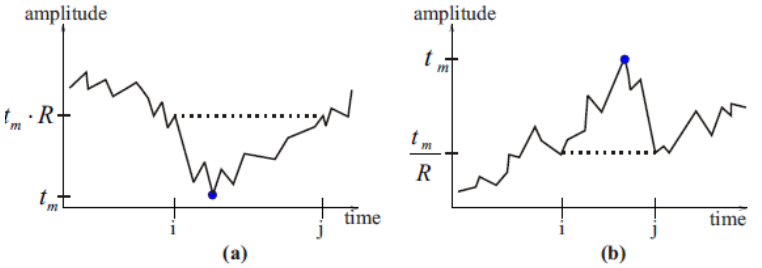

Fig. 2. Illustration of Significant Extreme Points: (a) Minimum, (b) Maximum

Notice that in the above definition, the parameter R is called *compression rate* which is greater than one and an increase of R leads to selection of fewer significant extreme points. Fig. 2 illustrates the definition of significant minima (a) and maxima

(b). Given a time series *T*, starting at the beginning of the time series, all significant minima and maxima of the time series are computed by using the algorithm given in [8].

The significant extreme points can be the starting point or ending point of a motif instances. Basing on the extracted significant points we can extract the motif candidates from a time series and then cluster them using BIRCH algorithm.

2.5 BIRCH Clustering

BIRCH is designed for clustering a large amount of numerical data by integration of hierarchical clustering at the initial stage and other clustering methods, such as iterative partitioning at the later stage ([13]). It introduces two main concepts, *clustering feature* and *clustering feature tree* (CF tree), which are used to summarize cluster representations. These structures help the clustering method achieve good speed and scalability in large databases. BIRCH is also effective for incremental and dynamic clustering of incoming objects.

Given *N d*-dimensional points or objects $\vec{x_i}$ in a cluster, we can define the centroid $\vec{x_0}$, the radius *R*, and the diameter *D* of the cluster as follows:

$$\vec{x_0} = \frac{\sum_{i=1}^{N} \vec{x_i}}{N}$$

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{x_i} - \vec{x_0})^2}{N} \right)^{\frac{1}{2}}$$

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{x_i} - \vec{x_j})^2}{N(N-1)} \right)^{\frac{1}{2}}$$

where *R* is the average distance from member objects to the centroid, and *D* is the average pairwise distance within a cluster. Both *R* and *D* reflect the tightness of the cluster around the centroid. A clustering feature (CF) is a triplet summarizing information about clusters of objects. Given *N d*-dimensional points or objects in a subcluster, then the CF of the cluster is defined as

$$\mathrm{CF} = (N, \overrightarrow{LS}, SS)$$

where *N* is the number of points in the subcluster, $\overrightarrow{LS}$ is the linear sum on *N* points and *SS* is the square sum of data points.

$$\overrightarrow{LS} = \sum_{i=1}^{N} \vec{x_i}$$

$$SS = \sum_{i=1}^{N} \vec{x}_i^{\,2}$$

A clustering feature is essentially a summary of the statistics for the given subcluster: the zero-th, first, and second moments of the subcluster from a statistical point of view. Clustering features are *additive*. For example, suppose that we have two disjoint clusters, C_1 and C_2, having the clustering features, CF_1 and CF_2, respectively. The clustering feature for the cluster that is formed by merging C_1 and C_2 is simply $CF_1 + CF_2$. Clustering features are sufficient for calculating all of the measurements that are needed for making clustering decisions in BIRCH.

A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering. By definition, a nonterminal node in the tree has descendents or "children". The nonleaf nodes store sums of the CFs of their children, and thus summarize clustering information about their children. Each entry in a leaf node is not a single data objects but a subcluster. A CF tree has two parameters: branching factor (B for nonleaf node and L for leaf node) and threshold T. The branching factor specifies the maximum number of children in each nonleaf or leaf node. The threshold parameter specifies the maximum diameter of the subcluster stored at the leaf nodes of the tree. The two parameters influence the size of the resulting tree.

BIRCH applies a multiphase clustering technique: a single scan of the data set yield a basic good clustering, and one or more additional scans can (optionally) be used to further improve the quality. The BIRCH algorithm consists of four phases as follows.

Phase 1: (*Building CF tree*) BIRCH scans the database to build an initial in-memory CF tree, which can be view as a multilevel compression of the data that tries to preserve the inherent clustering structure of the data.

Phase 2: [optional] (*Condense data*) Condense into desirable range by building a smaller CF tree.

Phase 3: (*Global Clustering*) BIRCH applies a selected clustering algorithm to cluster the leaf nodes of the CF tree. The selected algorithm is adapted to work with a set of subclusters, rather than to work with a set of data points.

Phase 4: [optional] *Cluster refining*

After the CF tree is built, any clustering algorithm, such as a typical partitioning algorithm, can be used in Phase 3 with the CF tree built in the previous phase. Phase 4 uses the centroids of the clusters produced by Phase 3 as seeds and redistributes the data points to its closest seed to obtain a set of new clusters.

3 The Proposed Method – Combination of Significant Extreme Points and BIRCH

The proposed method, called EP-BIRCH (Extreme points and BIRCH clustering), is an improvement of the EP-C described in Section 2. The EP-C algorithm by Gruber et al. [2] uses hierarchical agglomerative clustering (HAC) algorithm for clustering

which is not suitable to large scale time series datasets. In our proposed method, we use BIRCH algorithm to cluster motif candidates rather than using HAC algorithm. BIRCH is especially suitable for clustering very large time series datasets. Besides, in the EP-C algorithm, each motif candidate is determined by three contiguous extreme points, but in our proposed method, motif candidate is determined by n contiguous extreme points where n is selected by user.

EP-BIRCH consists of the following steps:

Step 1: We extract all significant extreme point of the time series T. The result of this step is a sequence of extreme points $EP = (ep_1, \ldots, ep_l)$

Step 2: We compute all the motif candidates iteratively. A motif candidate $MC_i(T)$, $i = 1, \ldots, l - 2$ is the subsequence of T that is bounded by the n extreme points ep_i and ep_{i+n-1}. Motif candidates are the subsequences that may have different lengths.

Step 3: Motif candidates are the subsequences that may have different lengths. To enable the computation of distances between them, we can bring them to the same length using homothetic transformation. The same length here is the average length of all motif candidates extracted in Step 2.

Step 4: We build the CF tree with parameters B and T. We insert to the CF tree all the motif candidates found in Step 3. We apply k-Means as Phase 3 of BIRCH to cluster the leaf nodes of the CF tree where k is equal to the number of the leaf nodes in the CF tree.

Step 5: Finally we find the subcluster in the CF tree with the largest number of objects. The 1-motif will be represented by that cluster.

In the Step 3, to improve the effectiveness of our proposed method, we apply *homothety* for transforming the motif candidates with different lengths to those of the same length rather than spline interpolation as suggested in [2]. Spline interpolation is not only complicated in computation, but also can modify undesirably the shapes of the motif candidates. Homothety is a simpler and more effective technique which also can transform the subsequences with different lengths to those of the same length.

Homothety is a transformation in affine space. Given a point O and a value $k \neq 0$. A homothety with center O and ratio k transforms M to M' such that $\overrightarrow{OM'} = k \times \overrightarrow{OM}$. Fig. 3. shows a homothety with center O and ratio k = ½ which transforms the triangle MNP to the triangle M'N'P'.

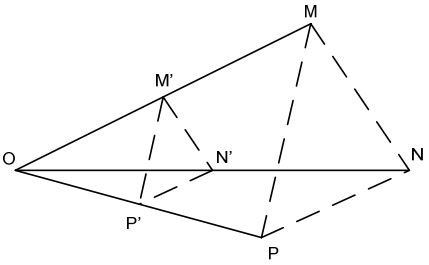

Fig. 3. Homothetic Transformation

Homothety can preserve the shapes of any curves under the transformation. Therefore, it can be used to align a longer motif candidate to a shorter one. The algorithm that performs homothety to transform a motif candidate T with length N ($T = \{Y_1,\ldots,Y_N\}$) to motif candidate of length N' is given as follows.

1. Let $Y\_Max = \text{Max}\{Y_1,\ldots,Y_N\}$; $Y\_Min = \text{Min}\{Y_1,\ldots,Y_N\}$
2. Find a center I of the homothety with the coordinate: $X\_Center = N/2$, $Y\_Center = (Y\_Max + \text{Y\_Min})/2$
3. Perform the homothety with center I and ratio $k = N'/N$.

Notice that in Step 4 of our proposed method, if the parameters B and T are well selected, the number of the leaf nodes in the CF tree is approximately the suitable number of the clusters for the particular set of motif candidates.

4 Experimental Evaluation

In this experiment, we compare our EP-BIRCH algorithm to the modified MK algorithm described in Section 2. The MK algorithm is selected for comparison since it is the most recent proposed motif discovery algorithm which has remarkable efficiency. We implemented the two motif discovery algorithms with Microsoft Visual C# and conducted the experiment on a Core i7, Ram 4GB PC. We tested the algorithms on six publicly available datasets. The datasets are described as follows.

1. Monthly air temperatures in Tokyo, measured at the Station No:47662, from 01/1876 to 06/2012[1]
2. Natural Gas Futures Contract 1 (Dollars per Million BTU) from 31/12/1993 to 13.07.2012[2].
3. Power Demand by ECN, displayed as a function of hours and days[3]
4. Euro/US Dollar Exchange rates from 28.03.2005 to 28.03.2006, measured at every 5 minutes[4]
5. Koski ECG (electrocardiogram) dataset[5]
6. Sea level dataset, measured at Coastal Ocean Observation Network TCOON, at every 6 minutes[6]

The comparison is in terms of running time and efficiency. Here we evaluate the efficiency of each algorithm by simply considering the ratio of how many times the Euclidean distance function must be called by this algorithm over the number of times it must be called by the brute-force algorithm given in Section 2. The efficiency value

[1] http://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3_en.php?block_no=47662&view=7
[2] http://www.eia.gov/dnav/ng/hist/rngc1d.htm
[3] http://www.cs.ucr.edu/~eamonn/Keogh_Time_Series_CDrom
[4] http://www.forexpros.com/currencies/eur-usd-historical-data
[5] http://www.cs.ucr.edu/~eamonn/iSAX/koski_ecg.dat
[6] http://lighthouse.tamucc.edu/pq

is always less than 1; the method with lower efficiency value is better. The experimental results for the efficiency of the two motif discovery algorithms, EP-BIRCH and MK, on the six datasets are shown in Table 1.

Table 1. Experimental results on the two algorithms over 6 datasets

Data	Length	Motif average length	Efficiency (%)		Runtime (sec)	
			MK	EP-BIRCH	MK	EP-BIRCH
Tokyo air tempetature	1639	13	9.16	1.537	5.807	0.382
Natural Gas	4638	34	33.64	5.569	12.789	1.002
Power	35040	99	2.43	0.112	120.741	3.015
Euro/USD	78893	36	0.7	0.13	532.023	13.383
ECG	144404	157	6.15	0.004	9146.328	4.344
TCOON	175200	112	Out of memory	0.046	Out of memory	21.167

From the experimental results in Table 1 we can see that:

1. EP-BIRCH is more efficient than MK in terms of CPU times and efficiency values. EP-BIRCH is up to four orders of magnitude faster than brute-force algorithm.
2. EP-BIRCH can find motifs on large time series dataset. With large datasets such as TCOON (175200 data points), MK can not work, while EP-BIRCH can find the motif in a very short time (21 seconds)
3. EP-BIRCH can find motif instances with different lengths.
4. The performance of EP-BIRCH is quite stable when some input parameters are changed.
5. When the input parameters are set with suitable values, the 1-motif found by EP-BIRCH is exactly similar to the 1-motif found by the bruce-force algorithm.

The Effects of Parameters on the Performance of EP-BIRCH

EP-BIRCH requires from user 4 parameters: R (compression rate for computing the significant extreme points, n (the number of significant extreme points for each motif candidate, B (the branching factor of a nonterminal node in CF tree) and T (the maximum diameter of the subclusters stored in the leaf nodes in CF tree). The length of motif candidates is determined by the two parameters R and n. With larger R, less extreme points are extracted and the distance between two extreme points will become larger. With smaller R, more extreme points are extracted and the distance between two extreme points will become shorter. However when we increase n we will obtain motif candidates with larger length. Therefore, we can determine easily the values of R and n such that we obtain the desirable motif length.

We conducted an experiment to compare the motifs detected by EP-BIRCH when we change the values of parameters T and B. Experiments on the ECG dataset shows the following results:

- For $n = 2$: When T changes from 0.4 to 1.4 and for all different values of B, all the detected motifs are the same.
- For $n = 3$: When T changes from 1.0 to 2.0 and for all different values of B, all the detected motifs are the same.

The experimental results reveal that EP-BIRCH is quite stable when the two parameters T and B change in some given ranges.

5 Conclusions

We have introduced a new method for discovering motifs in time series which can work efficiently on large time series datasets. This method, called EP-BIRCH, is based on extracting significant extreme points and clustering the motif candidates by using BIRCH algorithm. The experiments on the real world datasets demonstrate that our proposed method outperforms the MK algorithm in terms of efficiency. Notice that our proposed method requires only one single scan over the entire time series dataset. Therefore, we can apply EP-BIRCH not only for discovering motifs on large time series datasets, but also for finding motifs in streaming time series.

As for future work, we plan to extend the proposed method in finding motifs in streaming time series and create a disk-aware version of our algorithm to allow the exploration of truly massive time series datasets.

References

1. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In: Proc. of 9th Int. Conf. on Knowledge Discovery and Data Mining (KDD 2003), pp. 493–498 (2003)
2. Gruber, C., Coduro, M., Sick, B.: Signature verification with dynamic RBF network and time series motifs. In: Proc. of 10th International Workshop on Frontiers in Hand Writing Recognition (2006)
3. Li, Y., Lin, J.: Approximate Variable-Length Time Series Motif Discovery Using Grammar Inference. In: Proceedings of the Tenth International Workshop on Multimedia Data Mining, Washington, D.C (July 25, 2010)
4. Lin, J., Keogh, E., Patel, P. and Lonardi, S.: Finding Motifs in Time Series. In: Proceedings of the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002)
5. Liu, Z., Yu, J.X., Lin, X., Lu, H., Wang, W.: Locating Motifs in Time-Series Data. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 343–353. Springer, Heidelberg (2005)
6. Mueen, A., Keogh, E., Zhu, Q., Cash, S., Westover, B.: Exact Discovery of Time Series Motif. In: Proc. of 2009 SIAM International Conference on Data Mining, pp. 1–12 (2009)

7. Patel, P., Keogh, E., Lin, J., Lonardi, S.: Mining motifs in massive time series databases. In: Proc. of IEEE Int. Conf. on Data Mining, pp. 370–377 (2002)
8. Pratt, K.B., Fink, E.: Search for patterns in compressed time series. International Journal of Image and Graphics 2(1), 89–106 (2002)
9. Tang, H., Liao, S.: Discovering Original Motifs with Different Lengths from Time Series. Knowledge-based Systems 21(7), 666–671 (2008)
10. Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle. Machine Learning 58(2-3), 269–300 (2005)
11. Tin, H.: N: Time Series Motif Discovery based on Important Extreme Points, Master Thesis, Faculty of Computer Science and Engineering, Ho Chi Minh University of Technology, Vietnam (July 2012)
12. Yankov, D., Keogh, E., Medina, J., Chiu, B., Zordan, V.: Detecting Time Series Motifs Under Uniform Scaling. In: Proc. of 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2007), pp. 844–853 (2007)
13. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An efficient data clustering method for very large databases. SIGMOD Rec. 25(2), 103–114 (1996)