Left-Right Oscillate Algorithm for Community Detection Used in E-Learning System

Jan Martinovič[2], Pavla Dráždilová[1], Kateřina Slaninová[1], Tomáš Kocyan[2], and Václav Snášel[2]

[1] VŠB - Technical University of Ostrava,
Faculty of Electrical Engineering and Computer Science,
17. Listopadu 15/2172, 708 33 Ostrava, Czech Republic
{katerina.slaninova,pavla.drazdilova}@vsb.cz
[2] VŠB - Technical University of Ostrava,
IT4Innovations,
17. Listopadu 15/2172, 708 33 Ostrava, Czech Republic
{jan.martinovic,tomas.kocyan,vaclav.snasel}@vsb.cz

Abstract. Learning management systems are widely used as a support of distance learning. Recently, these systems successfully help in present education as well. Learning management systems store large amount of data based on the history of users' interactions with the system. Obtained information is commonly used for further course optimization, finding e-tutors in collaboration learning, analysis of students' behavior, or for other purposes. The partial goal of the paper is an analysis of students' behavior in a learning management system. Students' behavior is defined using selected methods from sequential and process mining with the focus to the reduction of large amount of extracted sequences. The main goal of the paper is description of our Left-Right Oscillate algorithm for community detection. The usage of this algorithm is presented on the extracted sequences from the learning management system. The core of this work is based on spectral ordering. Spectral ordering is the first part of an algorithm used to seek out communities within selected, evaluated networks. More precise designations for communities are then monitored using modularity.

1 Introduction

E-learning is a method of education which utilizes a wide spectrum of technologies, mainly internet or computer-based, in the learning process. It is naturally related to distance learning, but nowadays is commonly used to support face-to-face learning as well. *Learning management systems* (LMS) provide effective maintenance of particular courses and facilitate communication within the student community and between educators and students [9]. Such systems usually support the distribution of study materials to students, content building of courses, preparation of quizzes and assignments, discussions, or distance management of classes. In addition, these systems provide a number of collaborative learning tools such as forums, chats, news, file storage etc.

Regardless of LMS benefits, huge amount of recorded data in large collections makes often too difficult to manage them and to extract useful information from them. To

A. Cortesi et al. (Eds.): CISIM 2012, LNCS 7564, pp. 278–289, 2012.

overcome this problem, some LMS offer basic reporting tools. However, in such large amount of information the outputs become quite obscure and unclear. In addition, they do not provide specific information of student activities while evaluating the structure and content of the courses and its effectiveness for the learning process [26]. The most effective solution to this problem is to use data mining techniques [1].

The main goal of the paper is the description of our Left-Right Oscillate algorithm for community detection. The usage of this algorithm is presented on the extracted sequences from a learning management system. The core of this work is based on spectral ordering. Spectral ordering is the first part of an algorithm used to seek out communities within selected, evaluated networks. More precise designations for communities are then monitored using modularity.

The discovery and analysis of community structure in networks is a topic of considerable recent interest in sociology, physics, biology and other fields. Networks are very useful as a foundation for the mathematical representation of a variety of complex systems such as biological and social systems, the Internet, the world wide web, and many others [8,17]. A common feature of many networks is community structure, the tendency for vertices to divide into groups, with dense connections within groups and only sparser connections between them [12,18].

2 Analysis of Students' Behavior

Several authors published contributions with relation to mining data from e-learning systems to extract knowledge that describe students' behavior. Among others we can mention for example [14], where authors investigated learning process of students by the analysis of web log files. A 'learnograms' were used to visualize students' behavior in this publication. Chen et al. [3] used fuzzy clustering to analyze e-learning behavior of students. El-Hales [11] used association rule mining, classification using decision trees, E-M clustering and outlier detection to describe students' behavior. Yang et al. [25] presented a framework for visualization of learning historical data, learning patterns and learning status of students using association rules mining. The agent technology and statistical analysis methods were applied on student e-learning behavior to evaluate findings within the context of behavior theory and behavioral science in [2].

Our subject of interest in this paper is student behavior in LMS, which is recorded in form of events and stored in the logs. Thus, we can define the student behavior with the terms of process mining which are used commonly in business sphere. Aalst et al. [23,22] defines event log as follows:

Let A be a set of activities (also referred as tasks) and U as set of performers (resources, persons). $E = A \times U$ is the set of (possible) events (combinations of an activity and performer). For a given set A, A^* is the set of all finite sequences over A. A finite sequence over A of length n is mapping $\sigma =< a_1, a_2, \ldots, a_n >$, where $a_i = \sigma(i)$ for $1 \leq i \leq n$. $C = E^*$ is the set of possible event sequences. A simple event log is a multiset of traces over A.

The behavioral patterns are discovered using similarity of extracted sequences of activities performed in the system.

A sequence is an ordered list of elements, denoted $< e_1, e_2, \ldots, e_l >$. Given two sequences $\alpha =< a_1, a_2, \ldots, a_n >$ and $\beta =< b_1, b_2, \ldots, b_m >$. α is called a subsequence of β, denoted as $\alpha \subseteq \beta$, if there exist integers $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ such that $a_1 = b_{j1}, a_2 = b_{j2}, \ldots, a_n = b_{jn}$. β is than a super sequence of α.

For finding the behavioral patterns, we need to use the methods for the sequence comparison. There are generally known several methods for the comparison of two or more categorical sequences. On the basis of our previous work [21] we have selected the algorithm, witch deals with the different lengths of sequences and with the possible error or distortion inside the sequence.

Time-warped longest common subsequence (T-WLCS) [13] is the method, which combines the advantages of two methods from pattern mining - The longest common subsequence (LCSS) [15] and Dynamic time warping (DTW) [16]. LCSS allows us to find the longest common subsequence of two compared sequences. DTW is used for finding the optimal visualization of elements in two sequences to match them as much as possible. Then, selected T-WLCS method is able to compare sequences of various lengths, it takes into consideration the order of elements in the sequences, and it is immune to minor distortions inside of one of the compared sequences. Moreover, the method emphasizes recurrence of the elements in one of the compared sequences.

To obtain behavioral patterns of similar sequences in LMS, we have used our proposed new Left-Right Oscillate algorithm for community detection. The algorithm is based on spectral ordering (see Section 3).

3 Spectral Clustering and Ordering

Spectral clustering has become one of the most popular modern clustering algorithms in recent years. It is one of the graph theoretical clustering techniques and is simple to implement, can be solved efficiently by standard linear algebra methods, and very often outperforms traditional clustering algorithms such as the k-means or single linkage (hierarchical clustering). A comprehensive introduction to the mathematics involved in spectral graph theory is the textbook of Chung [5]. Spectral clustering algorithm uses eigenvalues and eigenvectors of Laplacian of similarity matrix derived from the data set to find the clusters. A practical implementation of the clustering algorithm is presented in [4]. Recursive spectral clustering algorithm is used in [6]. There Dasgupta et al. analyzed the second eigenvector technique of spectral partitioning on the planted partition random graph model, by constructing a recursive algorithm. A spectral clustering approach to finding communities in graphs was applied in [24].

Ding and He showed in [7] that a linear ordering based on a distance sensitive objective has a continuous solution which is the eigenvector of the Laplacian. Their solution demonstrate close relationship between clustering and ordering. They proposed direct K-way cluster assignment method which transforms the problem to linearization the clustering assignment problem. The linearized assignment algorithm depends crucially on an algorithm for ordering objects based on pairwise similarity metric. The ordering is such that adjacent objects are similar while objects far away along the ordering are dissimilar. They showed that for such an ordering objective function the inverse index

permutation has a continuous (relaxed) solution which is the eigenvector of the Laplacian of the similarity matrix.

3.1 Modularity-Quality of Detected Communities

To quantify the quality of the subdivisions we can use modularity [20], defined as the fraction of links between the nodes in the same community minus their expected value in a corresponding random graph [20]. Networks with the high modularity have dense connections between the nodes within community, but sparse connections between the nodes in the different communities. Modularity is often used in optimization methods for detecting community structure in the networks [19]. The value of the modularity lies in the range $\langle -0.5, 1\rangle$. It is positive, if the number of edges within groups exceeds the number expected on the basis of chance.

For a *weighted graph* G we have a weight function $w : E \to R$. It is for example function of the similarity between the nodes v_i and v_j. The weighted adjacency matrix of the graph is the matrix $W = (w_{ij})$ $i, j = 1, \ldots, n$. Than the degree of a vertex $v_i \in V$ in weighted graph is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}.$$

The weighted degree matrix D is defined as the diagonal matrix with the weighted degrees $d_1, \ldots, d_n$ on the diagonal.

In terms of the edge weights, modularity $Q(C_1, \ldots, C_k)$ is defined over a specific clustering into k known clusters $C_1, \ldots, C_k$ as

$$Q(C_1, \ldots, C_k) = \sum_{i=1}^{k} (e_{ii} - \sum_{j=1, i \neq j}^{k} e_{ij})$$

where $e_{ij} = \sum_{(u,v) \in E, u \in C_i, v \in C_j} w(u, v)$ with each edge $(u, v) \in E$ included at most once in the computation.

4 Left-Right Oscillate Algorithm for Community Detection

Upon completing our study of various modifications of algorithms for spectral clustering, we designed our own algorithm for detecting communities within complex networks. This algorithm utilizes spectral ordering where similar vertices are closer to indexes and less similar vertices are further from indexes. When determining the ordering, it is necessary to calculate the eigenvector of the second smallest eigenvalue of the matrix $L = D - W$. Since we have designed our algorithm for large amounts of data in a complex network, we used Lanczos method to calculate the Fiedler vector. Once the Fiedler vector was calculated, we detected appropriate gaps that divide the vertices of a graph into communities. As observed in the experiment, this type of separation into gaps leads to several badly-assigned subgraphs. This is due to the fact that the Fiedler vector is only linear ordered, as is revealed in our data collection. The Left-Right algorithm (see Algorithm 3.1) we have designed for incorporating small subgraphs into larger communities, gradually increases modularity in a given calculation.

Algorithm 1. Left-Right Algorithm for Community Detection

Input: similarity matrix $W = w_{i,j}$ for $i = 1,\ldots n$ and S_c size of smallest communities.
Output: communities C_k, modularity of detected communities

1. Create Laplacian $L = D - W$ using a matrix of similarity W of a connected graph $G = (V,E,W)$.
2. Calculate the Fiedler vector (the second eigenvector of Laplacian).
3. Reorder vertices according to Fiedler vector.
4. Calculate the sums of antidiagonals $Asum_i = \sum_{\forall j} w_{i-j,i+j}$ a $Asum_{(i\pm 1/2)} = \sum_{\forall j} w_{i-j,i+j\pm 1}$ for all $i = 1,\ldots n$ and determine $sum_i = Asum(i-1/2)/4 + Asum(i)/2 + Asum(i+1/2)/4$.
5. Approximate the discrete function sum_i by its spline and determine its first and second derivation. Then find all local minimums and maximums.
6. Assign maximum gaps that lie between two local maximums. Divide the set of vertices according to its gaps. Obtain subsets $SS_k \subset V$, where $k = 1,\ldots K$ is the amount of subsets.
7. Detect a community using the Left-Right Oscillate assigning algorithm (see Algorithm 2).

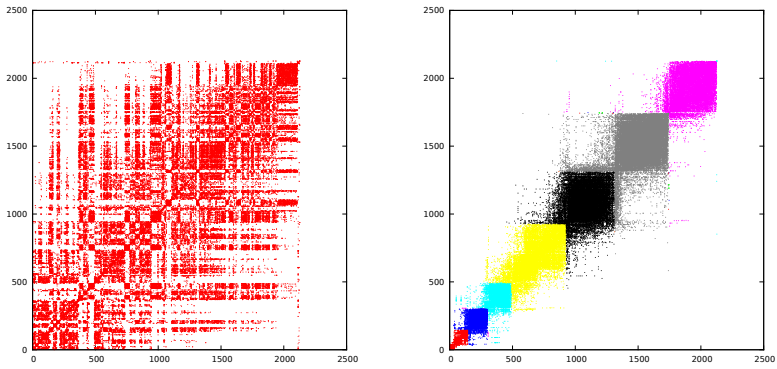

Fig. 1. Similarity Matrix and Permuted Similarity Matrix (Natural Number of Communities is 7)

Spectral ordering minimizes the sum of weighted edges multiplied to the power of the difference in index nodes with the edge incidence. The calculation used for this equation is the given eigenvector of the second smallest eigenvalue (Fiedler vector) matrix $L = D - W$. A visualized Fiedler vector and ordered matrix similarity (in agreement with the Fiedler vector) reveals the creation of several natural clusters which is assigned by our algorithm.

For finding the Fiedler vector of Laplacian above a large, sparse and symmetric matrix representative of the evaluated network, we used Lanczos method to partially solve the eigenvalue problem. To determine the dimension of Krylov subspaces (for a more precisely calculated Lanczos method), we used modularity for determining the quality of a detected community. In [7], there is an example of a symmetric Laplacian $L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. Our experiment revealed, that this solution is only appropriate for some networks.

Algorithm 2. Left-Right-Oscillate Assigned

Input: subsets $SS_k \subset V$, where $k = 1,\ldots K$, S_c size of smallest communities.
Output: communities C_k, modularity of detected partitioning.

1. Find connected components C_j for subset SS_1, which are greater than the selected size $|C_j| \geq S_c$. These components create communities. We add the rest of the vertices $v_i \in SS_1 - \bigcup C_j$ to the next subset of vertices SS_2.
2. Find next connected components C_j for every subset SS_k $k = 2,\ldots K$, which are greater than the selected size $|C_j| \geq S_c$. These components create communities. Attempt to assign other vertices to the previous community, which was established in the previous step. If the vertex has no edge leading to the previous community than we add the vertex to the next subset of vertices SS_{k+1}. Continue repeating this method 2 until reach the end of ordered vertices.
3. Going through through all subsets of vertices, connected components are assigned to C_j for $j = 1,\ldots J-1$ and C_J contain a set of connected components smaller than the selected size.
4. Employ the same approach going right-left without "oscillation". Begin with $C_{J-1} = C_{J-1} \cup C_J$.

The next step for the algorithm is to order indexes of vertices $v_i \in V$ for all $i = 1,\ldots n$ in compliance with ordering using Fiedler vector values. Because we want to find communities that are easily detected in a visual representation when ordered by a similarity matrix, we must determine where one community in a linear order ends and the next begins (find two nodes that belong to various communities). For this reason, we have calculated the value of antidiagonal sums above an ordered the set of vertices that capture a cluster overlap in neighboring vertices v_i. We define cluster crossing as the sum of a small fraction of the pairwise similarities. This is aided by linear ordering data points. The goal is to find the nodes that lay in the areas with fewer edges. These vertices lie close to locales with minimum function that are attached by approximation of a cluster overlap discrete function Sum_i. We assigned this approximation using the spline function, allowing for easy calculations of both the first and second derivation, which are used to assign local extremes. Between the two local maximum extremes of this function, there lie two vertices. In this area, these vertices represent a maximum gap (the difference in their Fiedler vector value). This gap determines the border between two potential communities.

Using this method, we have found the natural amount of 'communities' above a given evaluated network. Since the precision with which the Fiedler vector is calculated is a determining factor, and since in some cases vertices are incorrectly assigned, the result is an irrelevant component. The benefit of using our algorithm lies within its ability to assign isolated nodes (or very small subgraphs with selected sizes) to the nearest, most suitable, connected component that creates the nucleus of a future community. Within our assignments, we gradually arrive at a set of vertices V separated by gaps in individual subsets V_k. If the found set V_k does not create a connected subgraph ($G_k = (V_k, E)$, we determine all connected components in this subgraph. The maximum connected subgraph then creates the nucleus of this community and all subgraphs smaller than the selected size are moved to the right. We then attempt to reassign the subgraph to the

next subset of vertices V_{k+1}. Due to the linear nature of spectral ordering, it is presumable that subgraphs not yet assigned are reordered to the next subset of vertices. This means that we add the vertices of these subgraphs to the vertices of the next subset (that came into existence along gaps and creates a subgraph of the original graph with a set of vertices V_{k+1}). Then we test the connectivity of subgraph G_{k+1}, which was expanded by the nodes from the previous, unassigned subgraph. We go through the entire, spectrally ordered set of graph vertices employing this method. At the end of this process, we have created the most relevant of components within which we assign small subgraphs that are not yet assigned. Then, we repeat this approach in the opposite direction - going from right to left - and we try to add vertices for inspection in a subgraph. We may then assign the vertices to a connected subgraph with adjacency to a vertex of a given subgraph.

Once we assign a subset of vertices using gaps, and once we have detected connected components from left to right and vice versa, we always calculate the modularity for the obtained separation of graphs into subgraphs. Our results have revealed that our Left-right method increases modularity. The resulting connected subgraphs then create the structure of communities in the graph, which is demonstrated on well known data collection Zachary karate club in Table 1.

Table 1. Modularity Before and After Left-Right Algorithm for Zachary Karate Club

	Before Left-right	Commun.	After Left	Commun.	After Left-right	Commun.
Laplacian	0.272	11	0.361	4	0.361	4
Normalized-cut	0.342	7	0.311	4	0.311	4

5 Sequence Extraction in LMS Moodle

In this section is presented the extraction of students' behavioral patterns performed in the e-learning educational process. The analyzed data collections were stored in the Learning Management System (LMS) Moodle logs used to support e-learning education at Silesian University, Czech Republic.

The logs consist of records of all events performed by Moodle users, such as communication in forums and chats, reading study materials or blogs, taking tests or quizzes etc. The users of this system are students, tutors, and administrators; the experiment was limited to the events performed only by students.

Let us define a set of students (users) U, set of courses C and term *Activity* $a_k \in A$, where $A = P \times B$ is a combination of activity prefix $p_m \in P$ (e.g. course view, resource view, blog view, quiz attempt) and an action $b_n \in B$, which describes detailed information of an activity prefix (concrete downloaded or viewed material, concrete test etc.). *Event* $e_j \in E$ then represents the activity performed by certain student $u_i \in U$ in LMS. On the basis of this definition, we have created a set S_i of sequences s_{ij} for the user u_i, which represents the students' (users') paths (sessions) on the LMS website. *Sequence* s_{ij} is defined as a sequence of activities, for example $s_{ij} =< a_{1j}, a_{2j}, \ldots, a_{qj} >$, which is j-th sequence of the user u_i.

The sequences were extracted likewise the user sessions on the web; the end of the sequences was identified by at least 30 minutes of inactivity, which is based on our previous experiments [10]. Similar conclusion was presented by Zorrilla et al. in [26].

Using this method, we have obtained a set of all sequences $S = \cup_{\forall i} S_i$, which consisted of large amount of different sequences s_l performed in LMS Moodle. We have selected the course Microeconomy A as an example for the demonstration of proposed method. In Table 2 is presented detailed information about the selected course.

Table 2. Description of Log File for Course Microeconomy A

Records	Students	Prefixes	Actions	Sequences
65 012	807	67	951	8 854

Table 3. Description of Sequence Graphs for T-WLCS Method

T-WLCS				
θ	**Isolated Nodes**	**Edges**	**Avg. Degree**	**Avg. Weighted Degree**
0.1	31	5577366	944.036	179.431
0.2	**143**	**1739042**	**294.354**	**88.883**
0.3	606	534648	90.496	38.735
0.4	**1200**	**271826**	**46.010**	**22.998**
0.5	2465	103028	17.439	10.430
0.6	3781	29298	4.959	3.596
0.7	**5038**	**8080**	**1.368**	**1.269**
0.8	5517	5914	1.001	0.997
0.9	5568	5788	0.980	0.980

The obtained set S of sequences consisted of large amount of different sequences, often very similar. Such large amount of information is hard to clearly visualize and to present in well arranged way. Moreover, the comparison of users based on their behavior is computationally expensive with such dimension. Therefore, we present in the article [21] the identification of significant behavioral patterns based on the sequence similarity, which allows us to reduce amount of extracted sequences.

We have used T-WLCS methods for the similarity measurement of sequences. The T-WLCS find the longest common subsequence α of compared sequences β_x and β_y, where $\alpha \subseteq \beta_x \wedge \alpha \subseteq \beta_y$, with relation to T-WLCS. Similarity was counted by the Equation 1.

$$Sim(\beta_x, \beta_y) = \frac{(l(\alpha) * h)^2}{l(\beta_x) * l(\beta_y)}, \tag{1}$$

where $l(\alpha)$ is a length of the longest common subsequence α for sequences β_x and β_y; $l(\beta_x)$ and $l(\beta_y)$ are analogically lengths of compared sequences β_x and β_y, and

$$h = \frac{Min(l(\beta_x), l(\beta_y))}{Max(l(\beta_x), l(\beta_y))} \tag{2}$$

On the basis of selected T-WLCS method for finding the similarity of sequences, we have constructed the similarity matrix for sequences ($|S| \times |S|$) which can be represented using tools of graph theory. For the visualization of network was constructed weighted graph $G(V,E)$, where weight w is defined as function $w : E(G) \rightarrow R$, when $w(e) > 0$. Set V is represented by set of sequences S, weights w are evaluated by the similarity of sequences, see Equation 1, depending on selected method. In Table 3 is more detailed description of weighted graphs of sequences, where weight is defined by T-WLCS method for selected threshold θ (treshold for edges filtering - edges with smaller weights are removed). The number of nodes for each graph is 5908.

5.1 Reduction of Large Amount of Sequences by Left-Right Oscillate Algorithm

We described the procedure for extraction of sequencec from the LMS system in previous parts of the paper. We created the graphs of sequences by T-WLCS method. The examples in this section show how the graphs of sequences are divided to clusters by Left-Right Oscillate algorithm. We will use the concept "clusters" instead of "communities" in this part of the paper because we used Left-Right Oscillate algorithm for finding clusters of sequences.

Table 4. Description of Selected Biggest Clusters of Sequence Graphs for T-WLCS Method

T-WLCS		
θ	**Nodes**	**Edges**
0.2	5763	1739040
0.4	4639	271732
0.7	142	1030

Table 5. Partitioning of Sequence Graph for $\theta \leq 0.2$

$S_c = 1$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.23471	85	560	365	104
Left-Right	0.23471	85	560	365	104
$S_c = 3$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.23471	85	560	365	104
Left-Right	**0.23721**	**7**	**581**	**421**	**104**
$S_c = 6$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.23471	85	560	365	104
Left-Right	**0.23717**	**5**	**580**	**424**	**104**

Table 6. Partitioning of Sequence Graph for $\theta \leq 0.4$

$S_c = 1$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.52153	656	829	648	497
Left-Right	0.52153	656	829	648	497
$S_c = 3$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.52153	656	829	648	497
Left-Right	**0.52820**	**101**	**956**	**720**	**579**
$S_c = 6$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.52153	656	829	648	497
Left-Right	**0.52955**	**59**	**962**	**739**	**594**
$S_c = 10$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.52153	656	829	648	497
Left-Right	**0.52890**	**40**	**952**	**794**	**586**

Table 7. Partitioning of Sequence Graph for $\theta \leq 0.7$

$S_c = 1$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.26457	63	15	15	14
Left-Right	0.26457	63	15	15	14
$S_c = 3$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.26457	63	15	15	14
Left-Right	**0.38731**	**12**	**23**	**22**	**21**
$S_c = 6$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.26457	63	15	15	14
Left-Right	**0.46304**	**7**	**45**	**15**	**14**
$S_c = 10$					
Modularity Type	**Modularity**	**Clusters**	**Size of 1th**	**Size of 2nd**	**Size of 3rd**
Original	0.26457	63	15	15	14
Left-Right	**0.45416**	4	**58**	**42**	**27**

For the illustration, we have selected three different graphs from Table 3. These are the graphs created with parameter θ is greater than 0.2, 0.4 and 0.7. In each graph, there was identified the largest connected component, and on the basis on this component was created the new graph. This graph was partitioned by our new Left-Right Oscillate algorithm (maximum cycles of Lanczos algorithm inside Left-Right Oscillate algorithm was set to 1500). The sizes of these newly generated graphs are presented in Table 4.

Individual outputs and quality cuts of graphs after Left-Right Oscillate algorithm can be seen in Table 5, Table 5 and Table 7. In these tables we have column "Modularity Type" where row "Original" is without applied Left-Right Algorithm and "Left-Right" is row with information after Left-Right algorithm. Other columns in these tables are "Modularity" (see section 3.1), "Clusters" with amount of clusters after partitioning and columns with sizes of the top three communities (columns "Size of 1th", "Size of 2th", "Size of 3th").

It is apparent that the modularity is improved, if the parameter S_c (size of smallest clusters - see Algorithm 3.1) of the Left-Right Oscillate algorithm is greater than 1.

6 Conclusion

In the paper we introduced the Left-Right Oscillate algorithm, which allows us to improve the results of community detection based on spectral ordering. We showed effect of parameter S_c on the quality of clustering of sequences, which were extracted from the Moodle e-learning system. This allows us to better identify the same behavior of students in the online e-learning system. Modularity was used for measuring the quality of the distribution of sequences within clusters. In the future work we want to consider using the Left-Right Oscillate algorithm for hierarchical graph of sequences partitioning. Thanks to this we want to aim a more appropriate division of student's behavioral patterns.

Acknowledgment. This work was partially supported by SGS, VSB – Technical University of Ostrava, Czech Republic, under the grant No. SP2012/151 Large graph analysis and processing and by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

References

1. Castro, F., Vellido, A., Nebot, A., Mugica, F.: Applying Data Mining Techniques to e-Learning Problems. In: Jain, L., Tedman, R., Tedman, D. (eds.) Evolution of Teaching and Learning Paradigms in Intelligent Environment. SCI, vol. 62, pp. 183–221. Springer, Heidelberg (2007)
2. Chen, B., Shen, C., Ma, G., Zhang, Y., Zhou, Y.: The evaluation and analysis of student e-learning behaviour. In: IEEE/ACIS 10th International Conference on Computer and Information Science (ICIS), pp. 244–248 (2011)
3. Chen, J., Huang, K., Wang, F., Wang, H.: E-learning behavior analysis based on fuzzy clustering. In: Proceedings of International Conference on Genetic and Evolutionary Computing (2009)
4. Cheng, D., Kannan, R., Vempala, S., Wang, G.: On a recursive spectral algorithm for clustering from pairwise similarities. Technical report, MIT (2003)
5. Chung, F.R.K.: Spectral Graph Theory, vol. 92. American Mathematical Society (1997)
6. Dasgupta, A., Hopcroft, J., Kannan, R., Mitra, P.: Spectral Clustering by Recursive Partitioning. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 256–267. Springer, Heidelberg (2006)
7. Ding, C., He, X.: Linearized cluster assignment via spectral ordering. In: Twentyfirst International Conference on Machine Learning, ICML 2004, vol. 21, p. 30 (2004)

8. Dorogovtsev, S.N., Mendes, J.F.F.: Evolution of Networks: From Biological Nets to the Internet and WWW, vol. 57. Oxford University Press (2003)
9. Dráždilová, P., Obadi, G., Slaninová, K., Al-Dubaee, S., Martinovič, J., Snášel, V.: Computational Intelligence Methods for Data Analysis and Mining of eLearning Activities. In: Xhafa, F., Caballé, S., Abraham, A., Daradoumis, T., Juan Perez, A.A. (eds.) Computational Intelligence for Tech. Enhanced Learning. SCI, vol. 273, pp. 195–224. Springer, Heidelberg (2010)
10. Dráždilová, P., Slaninová, K., Martinovič, J., Obadi, G., Snášel, V.: Creation of students' activities from learning management system and their analysis. In: Abraham, A., Snášel, V., Wegrzyn-Wolska, K. (eds.) IEEE Proceedings of International Conference on Computational Aspects of Social Networks, CASON 2009, pp. 155–160 (2009)
11. El-halees, A.: Mining students data to analyze learning behavior: a case study (2008)
12. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences of the United States of America 99(12), 7821–7826 (2002)
13. Guo, A., Siegelmann, H.: Time-Warped Longest Common Subsequence Algorithm for Music Retrieval, pp. 258–261. Universitat Pompeu Fabra (2004)
14. Hershkovitz, A., Nachmias, R.: Learning about online learning processes and students' motivation through web usage mining. Interdisciplinary Journal of E-Learning and Learning Objects 5, 197–214 (2009)
15. Hirschberg, D.S.: Algorithms for the longest common subsequence problem. J. ACM 24, 664–675 (1977)
16. Müller, M.: Information Retrieval for Music and Motion. Springer (2007)
17. Newman, M.E.J., Barabási, A.-L., Watts, D.J.: The structure and dynamics of networks, vol. 107. Princeton University Press (2006)
18. Newman, M.E.J.: Detecting community structure in networks. The European Physical Journal B Condensed Matter 38(2), 321–330 (2004)
19. Newman, M.E.J.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences of the United States of America 103(23), 8577–8582 (2006)
20. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E - Statistical, Nonlinear and Soft Matter Physics 69(2 Pt 2), 16 (2004)
21. Slaninová, K., Kocyan, T., Martinovič, J., Dráždilová, P., Snášel, V.: Dynamic time warping in analysis of student behavioral patterns. In: Proceedings of the Dateso 2012, Annual International Workshop on DAtabases, TExts, Specifications and Objects, pp. 49–59 (2012)
22. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st edn. Springer, Heidelberg (2011)
23. van der Aalst, W.M.P., Reijers, H.A., Song, M.: Discovering social networks from event logs. Comput. Supported Coop. Work 14(6), 549–593 (2005)
24. White, S., Smyth, P.: A spectral clustering approach to finding communities in graphs. In: Proceedings of the Fifth SIAM International Conference on Data Mining, vol. 119, p. 274 (2005)
25. Yang, F., Shen, R., Han, P.: Construction and application of the learning behavior analysis center based on open e-learning platform (2002)
26. Zorrilla, M.E., Menasalvas, E., Marín, D., Mora, E., Segovia, J.: Web Usage Mining Project for Improving Web-Based Learning Sites. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) EUROCAST 2005. LNCS, vol. 3643, pp. 205–210. Springer, Heidelberg (2005)