User Behavioural Patterns and Reduced User Profiles Extracted From Log Files

Kateřina Slaninová
Department of Informatics
SBA, Silesian University in Opava
Karviná, Czech Republic
Email: slaninova@opf.slu.cz

***Abstract*—This paper is focused on log files where one log file attribute is an originator of the recorded activity (originator is a person in our case). Hence, based on the similar attributes of people, we are able to construct models which explain certain aspects of a persons behaviour. Moreover, we can extract user profiles based on behaviour and find latent ties between users and between different user groups with similar behaviours. We accomplish this by our new approach using the methods from log mining, business process analysis, complex networks and graph theory. The paper describes the whole process of the approach from the log file to the user graph. The main focus is on the step called 'The finding of user behavioural patterns'.**

***Keywords*-analysis of users' behaviour; user profiles; behavioural patterns; complex networks**

I. Introduction

This paper is focused on analysis of users behaviour in the systems. The analysis is performed using methods from various research domains like log mining, business process analysis, complex networks, graph theory, artificial intelligence, and datamining. The main goal of this paper is to propose a novel methodology for the analysis of users behaviour within the systems which store their activity in log files.

The main idea of the proposed methodology is to find an approach which enables us to compare the users who behave in the system in the similar way. The proposed solution consists in description of the users by their user profiles. The user profiles are constructed on the basis of user activities (sequences of activities) performed by him/her in the analysed system. For example, it is possible to extract information such as users' behaviour on a website, customers' behaviour at an eshop, employees' behaviour in an enterprise system (ERP) or other similar information in other monitored systems using relevant information and timestamps.

The paper describes the whole process of the approach from the log file to the user graph. The approach was partially presented in our previous works [1], [2], [3]. The main focus of this paper is on the step called 'The finding of user behavioural patterns', see Figure 1. The organisation of the paper is as follows. At first, a basic principles from log analysis which are essential for the proposed methodology are mentioned in Section II. Section III describes the whole overview of our new approach including the introduction of the main steps. A detailed description of each step is presented in Section IV, including the part focused on finding user behavioural patterns and discussion. Finally, Section V contains a conclusion.

II. Log Analysis

Log analysis is a data mining process focused on the analysis of computer-generated records (also called audit trail records, event logs or transaction logs) [4]. A log file is usually a simple text file generated by a device, software, application or system. It consists of messages, which are represented by records of events performed in the system. Usually, event log files are created from other data sources such as databases, flat files, message logs, transaction logs, spread sheets, and etc.

A log typically consists of activity information and event (or process instance), and mostly of timestamp, originator (performer) and other necessary data. For example, a typical web log is stored by web server software to record the requests of the web client to the web server and its responses (we can assume the activities of visitors on a web site from recorded information). This type of log file commonly uses a standardised format and includes the following information: IP address of a client accessing the web page, user name, date and time of request, resource requested size in bytes of the data returned to the client and URL that referred the client to the resource. However, the web activities stored in logs may contain additional information. This depends on the system or the application making the records.

Let us assume that an event log purely contains data related to a single process. Each event in the log file refers to a single process instance, called a case. Aalst et al. [5], [6] provided the following assumptions about event logs:

- A *process* consists of *cases*.
- A case consists of *events* such that each event relates to precisely one case.
- Events within a case are ordered.
- Events can have *attributes*.

978-1-4799-3516-1/13/$31.00

III. From Log File to Users' Network: Overview of Our Approach

Our approach proceeds from the original social network approach with a modification focused on users' behaviour. The modification is based on a definition of the relationship between users. The original approach into the analysis of social networks deals with the assumption that the social network is a set of people (or groups of people) with social interactions among themselves [7]. Social interaction is commonly defined as an interaction between actors, such as communication, personal knowledge of each other, friendship and membership etc.

The modification extends the original approach of social network analysis by the perspective of the complex networks [8]. This type of view differs from the original approach due to the description of the relations between nodes (users, actors). The relation between the users is defined by their common attributes, characterising their behaviour in the system. More specifically, user behaviour in the system is defined by user profiles.

The user profiles are extracted using the methods from process mining [6]. Let us assume that an event log from the analysed system contains data related to activities executed by the users in the system. Therefore, in accordance with the Aalst definitions [5], [6], we can extract the following information from the log file:

- *events* - an event $e \in E$, where E is a set of all events, is represented by the row in the log file. Events can be characterised by attributes like $\#_{resource}(e)$, $\#_{time}(e)$, $\#_{activity}(e)$, and others. Each event appears in the event log only once.
  - *resources* - a resource (performer) of an event $\#_{resource}(e)$ is an originator which performs the event e in the system. The resource can usually be a device, software or a person. In the our approach, the main focus is aimed towards users as a resource (students in LMS, agents in multi-agent system, users of the website). Resources can have additional attributes, for example the ability to sell (agents in multi-agent system).
  - *time* - each event in the presented approach should have an attribute $\#_{time}(e)$, which is the timestamp of the event e. This attribute represents the date and time when the event was executed. This information is essential for the extraction of traces (sequences of events).
  - *activities* - an activity $\#_{activity}(e)$ is an action associated with an event e. An activity is also called a labelled event in business processing. Activities may have additional attributes, for example concrete study material, quiz, type of contribution in a forum (LMS) or concrete material, URL address and website (web analysis).
- *cases* - an event log consists of cases $ca \in CA$, while cases consist of events. Each case has a special mandatory attribute trace $\#_{trace}(ca)$ and other optional attributes. In the presented approach, the cases can be related to a concrete performer $\#_{resource}(ca)$ (student, agent, web user), to a concrete study course $\#_{course}(ca)$ (LMS), to an agent negotiation $\#_{neg}(ca)$ (multi-agent system) or to a web session $\#_{session}(ca)$ (web analysis). Cases are uniquely identifiable. Other attributes (for example in multi-agent system) can be a price, an amount of items etc.
- *traces* - a trace $\#_{trace}(c)$ is a finite sequence of unique events $\sigma \in E$ such that for $1 \leq i < j \leq |\sigma| : \sigma(i) \neq \sigma(j)$. The extraction of the traces from the log file depends on the system generated the analysed log and on the type of data collection. In process mining and the analysis of business processes, traces are usually related to a business case during their construction (for example negotiation). However, in the our approach, a trace is related to a performer as well. This is due to the aim to construct the users' network on the basis of their behaviour in the system. In the presented approach a trace is named as a *sequence* of events which are more suitable for the data domain they are dealing with.

The main goal of our approach is to propose a methodology for an analysis of users' behaviour in the systems that store their activity in log files and the visualisation of latent ties among the groups of users with similar behaviour. This is done by using their profiles based on behavioural patterns. The main concept is described in Figure 1 and is shown with more details in the following text. User profiles are constructed using extracted sequences of events performed by users in the system from the log files (Figure 1, part a).

However, in some systems, the amount of extracted sequences can be very large (for example in LMS or in web server up to tens of thousands). Moreover, the sequences are often very similar. In such cases, user profiles described by a large amount of sequences (Figure 1, Base User Profiles, part b) are not trivial to compare in order to find the groups of users with similar behaviour using common methods for data clustering.

This suggests the need for the use of mathematical tools and the development of data reduction procedures by string alignment. Therefore, one step of the approach is focussed on the construction of reduced user profiles using these methods and the methods for sequence comparison. This step is focused on finding behavioural patterns (Figure 1, Sequence Reduction, part c), which are a base for the construction of new reduced user profiles (Figure 1, Reduced User Profiles, part d). Finally, the reduced user profiles facilitate the construction of a user network (Figure 1, part e) and the finding of user groups with similar behaviours.

Figure 1 shows a detailed approach with a view to the sequence reduction and discovering the behavioural patterns.

The process of sequence reduction is described in part c. The large set of extracted sequences S is mapped into a set of behavioural patterns R, where $|S| >> |R|$.

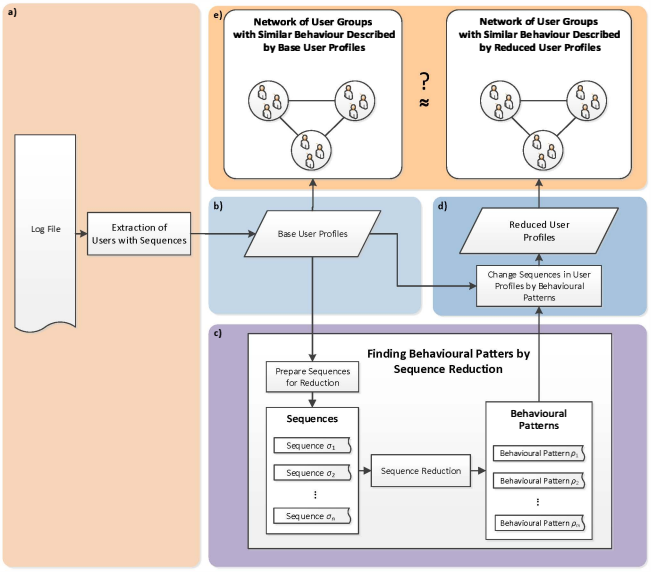

Figure 1. Detailed Approach

Our approach described in Figure 1 requires the solution of the following sub problems:

a) The definition of user behaviour in the system
b) The creation of user profiles based on their behaviour within the system
c) The finding of user behavioural patterns
d) The construction of user profiles based on user behavioural patterns
e) The finding of user groups with similar behaviours by using the behavioural patterns and the transparent visualisation of latent ties between them

The subproblems focused on definition of user behaviour in the system, creation of user profiles and finding of user groups with similar behaviours (steps a), b), and e) of the proposed approach) were described in detailes in our previous work [9]. This paper is focused on the finding of user behavioural patterns and on the construction of user profiles based on user behavioural patterns (steps c) and d) of the proposed approach). Both steps are presented in Section IV.

IV. User Behavioural Patterns and Reduced User Profiles

In [6], Aalst defines an event attribute transaction type $\#_{trans}(e)$. In the presented approach the transaction type is not known a priori. Therefore, it cannot be used as a label for the events. However, one of the aims of the approach is to discover this attribute. This is because finding the type of transaction (type of sequence) allows us to obtain *behavioural patterns* of the users in the system, see Definition 1. For this purpose, sequences σ_{ij} extracted with relation to certain user u_i are mapped to a set of sequences σ_l without this relation to users: $\sigma_{ij} = \langle e_{ij1}, e_{ij2}, \ldots, e_{ijm_j} \rangle \rightarrow \sigma_l \langle e_1, e_2, \ldots, e_{ml} \rangle$, where $e_{ij1} = e_1, e_{ij2} = e_2, \ldots, e_{ijm_j} = e_{ml}$.

Definition 1: (Representative, behavioural pattern): Let C be a set of clusters consisted of similar event sequences. Let $R = \{\rho_1, \rho_2, \ldots, \rho_r\}$ be a set of representatives of clusters ρ_k, where k=1, 2, …, r. Then, a representative $\rho_k \in R$ of a cluster C_k is a sequence, which describes a similar event sequences $\sigma_l = \langle e_1, e_2, \ldots e_{ml} \rangle$ in the cluster C_k (it has the maximal average similarity to the all sequences in the cluster C_k). Representatives $\rho_k \in R$ are called behavioural patterns.

Then, behavioural patterns were used for the creation of *reduced user profiles*, see Definition 2.

Definition 2: (Reduced user profile): Let U be a set of users u_i. Thus, a set $\Pi_i = \{\pi_1, \pi_2, \ldots \pi_{s_i}\}$ is a set of all behavioural patterns $\pi_k \in R$ for user u_i to which is mapped a set of sequences S_i executed by the user u_i in the system. A sequence σ_{ij} is mapped into such behavioural pattern π_k, which is a representative of a cluster to which belongs the corresponding sequence σ_{ij}. A reduced user profile of the user $u_i \in U$ is a vector $p_i \in N^{|R|}$ represented by row i from matrix P.

Define matrix $P \in N^{|U| \times |R|}$ where:

$$P_{ij} = \begin{cases} frequency\ of\ representative\ \rho_j \\ \qquad for\ user\ u_i \in R & \text{if } \rho_j \in \Pi_i \\ 0 & \text{else} \end{cases}$$

A reduced user profile of the user $u_i \in U$ is a vector $p_i \in N^{|R|}$ represented by row i from matrix P.

The solution of the problem of *The finding of user behavioural patterns*, consists of the three parts:

1) The determination of relation between sequences
2) The extraction of clusters with similar sequences
3) The determination of representatives for each cluster

A. Determination of Relations between Sequences

The relation between sequences was defined using methods for the comparison of the sequences LCS, LCSS or T-WLCS ([10], [11], [12]). These methods find the longest common subsequence (or the sequence which characterises the relation) z for comparing the sequences x and y. Then, the relation weight $w_{seq}(x, y)$ between sequences x and y was counted by Equation 1.

The methods LCS and LCSS used for the comparison of sequences find the longest common subsequence z of compared sequences x and y, where $(z \subseteq x) \wedge (z \subseteq y)$. The relation weight $w_{seq}(x, y)$ between the sequences x and y was counted by Equation 1:

$$w_{seq}(x, y) = \frac{l(z)^2}{l(x)l(y)} \frac{Min(l(x), l(y))^2}{Max(l(x), l(y))^2}, \qquad (1)$$

where $l(x)$ and $l(y)$ are lengths of the compared sequences x and y, and $l(z)$ is a length of a subsequence z. Equation 1 takes account of the possible difference between $l(x)$ and $l(y)$. Due to this reason, z is adapted so that $w_{seq}(x, y)$ is strengthened in the case of similar lengths of sequences x and y, and analogically weakened in the case of higher difference of $l(x)$ and $l(y)$. For the methods LCS and LCSS, w_{seq} meets all the similarity conditions: $w_{seq} \geq 0$, $w_{seq}(x, x) = 1$, $w_{seq}(x, x) > w_{seq}(x, y)$ and $w_{seq}(x, y) = w_{seq}(y, x)$.

The output z is only the sequence which characterizes the relation between the sequences x and y for T-WLCS method. Therefore, $w_{seq}(x, y)$ does not meet all the similarity conditions due to its characteristics. Respectively, it is possible that $w_{seq}(x, y) > w_{seq}(x, x)$. Although we know that $w_{seq}(x, y)$ is not a similarity for T-WLCS method, due to a simplification, the 'sequence similarity' will be used as a relation weight $w_{seq}(x, y)$ between the sequences x and y for all the methods of sequence comparison in the following text of this paper.

After counting the weights between the sequences, the undirected weighted graph $G(V, E)$ was obtained as the output, where vertices V (graph nodes) were represented by set of sequences S extracted from the log file. The edges, E, represented the relations between the sequences. The output was influenced by two parameters. The first parameter was the selected method (LCS, LCSS, T-WLCS, etc.) which was used for determining the weight w_{seq}. The second parameter was the minimal weight between the sequences which was used for the construction of the basic graph of sequences. If the weight was lower than this value then the relation between the two sequences was not created in the graph.

B. Extraction of Clusters with Similar Sequences

The extraction of the clusters with similar sequences was realized by spectral clustering. Spectral clustering is one of the divisive clustering algorithms which can be applied in the graph theory. The spectral clustering algorithm uses eigenvalues and eigenvectors of a similarity matrix derived from the data set to find the clusters. The usage of spectral algorithm was studied in [13] by Cheng et al. The practical implementation of the clustering algorithm is presented in [14]. The spectral clustering method optimised by our proposed Left-Right Oscillate algorithm [15], [3] was used in this paper.

The spectral clustering method, which is hierarchically applied on weighted and undirected graphs, leads to obtaining clusters which are often near to a complete graph. Due to this fact it is possible, in some cases, that the sequences with the high value w_{seq} may appear in one cluster w_{seq}. However, even the sequences, which are not similar at all, are always similar to their neighbour sequences.

This problem was solved by an algorithm for finding the most suitable representative sequence of the cluster. The algorithm allows an additional, and a more appropriate division of the clusters, if it is necessary for a selected similarity threshold θ, see Algorithm 1.

Algorithm 1 Determination of Representative in Cluster of Sequences

Goal: The determination of sequence representatives ρ (behavioural patterns) which will be used for a description of the sequence clusters C, for selected similarity threshold θ.

Input:

- A cluster C, where a similarity between the objects inside the cluster is given by values $Sim(c_l, c_k)$.
- A selected threshold θ for a sequence similarity w_{seq} in clusters.

Output:

- A set of representatives $R = \{\rho_1, \rho_2, \ldots, \rho_r\}$ and a set of clusters C_ρ which correspond to the appropriate representatives.

1) For each object, belonging into a cluster C, determine its average similarity to other objects inside that cluster. Similarity between the objects c_l and c_k is given by the condition $Sim(c_l, c_k) > \theta; \forall c_l, c_k \in C$, where θ is the selected threshold for the object similarity.
2) Select the object with the maximum average similarity counted in Step 1. The selected object is labelled ρ.
3) Create a new cluster C_ρ, which contains the objects from cluster C, which meets the condition $Sim(c_l, c_k) > \theta$ and contains the representative ρ.
4) Add a representative ρ into the set R and record the appropriate cluster C_ρ, which is created in Step 5.
5) $C = C - C_\rho$.
6) If $C \neq \emptyset$, then repeat to Step 1 with a new cluster C.

As mentioned before, the extraction of clusters with similar sequences was performed by spectral clustering. The output consisted of sets of clusters, which represented subgraphs with the similar sequences, for each level of hierarchy. Afterwards, further processing was done with the clusters on the last level of hierarchy (but it is possible to use higher levels with rough division of sequence clusters). Table I shows one of the clusters with similar sequences as found by spectral clustering; the similarity of sequences was defined by LCS method. As we can see, the cluster contained not only the similar sequences (ID 19, 21, 17), but even the sequences which had a low weight of similarity w_{seq} (ID 12 and 11) to the selected representative of the cluster.

C. Determination of representatives for each cluster

The next step of this application was a more precise division of the obtained sequence clusters C_ρ and finding the representatives for these obtained clusters using our Algorithm 1. As a partial output of this algorithm the clusters and their representatives was generated.

ID	Levenshtein Distance	LCS w_{seq}	Sequence Description
9	17	0.075	1 2 3 4 5 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 10
18	8	0.189	2 3 4 5 7 8 7 8 7 8 7 8 7 8 7 8 7 8 12 10
10	16	0.083	2 3 4 5 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 10
21	1	0.813	1 2 3 4 5 7 8 7 8 7 8 7 8 12 10
20	4	0.346	2 3 4 5 7 8 7 8 7 8 7 8 7 8 7 8 12 10
19	0	1.000	2 3 4 5 7 8 7 8 7 8 7 8 7 8 12 10
17	2	0.437	2 3 4 5 7 8 7 8 7 8 12 10
15	4	0.233	2 3 4 5 7 8 7 8 12 10
14	2	0.492	2 3 4 5 7 8 7 8 7 8 7 8 7 8 12 10
16	7	0.118	1 2 3 4 5 7 8 12 10
13	6	0.105	2 3 4 5 7 8 12 10
12	13	0.001	1 2 9 10
11	12	0.001	2 9 10

Table I
EXAMPLE OF SEQUENCE CLUSTER FOUND BY SPECTRAL CLUSTERING, LCS METHOD

In Example 4.1, there is shown a more appropriate division of the cluster presented in Table I. It was divided in smaller clusters of sequences using our proposed Algorithm 1 for threshold $\theta = 0.2$. As we can see, each cluster contains an ID of its representative sequence (best sequence) and a list of sequences in the clusters. Each sequence has its sequence ID, its similarity w_{seq} to the representative, and sequence description. The proposed algorithm has divided the founded cluster into four subclusters, where the sequences have a higher similarity.

Example 4.1 (Sequence Clusters and Representatives):

```
Best sequence: 19
Seq.: 21, lev: 1 sim: 0.813, data: 1 2 3 4 5 7 8 7 8 7 8 7 8
                                                    12 10
Seq.: 20, lev: 4 sim: 0.346, data: 2 3 4 5 7 8 7 8 7 8 7 8 7
                                                  8 7 8 12 10
Seq.: 19, lev: 0 sim: 1.000, data: 2 3 4 5 7 8 7 8 7 8 7 8
                                                    12 10
Seq.: 17, lev: 2 sim: 0.437, data: 2 3 4 5 7 8 7 8 7 8 12 10
Seq.: 15, lev: 4 sim: 0.233, data: 2 3 4 5 7 8 7 8 12 10
Seq.: 14, lev: 2 sim: 0.492, data: 2 3 4 5 7 8 7 8 7 8 7 8 7
                                                    8 12 10
Avrg. Levenshtein Distance: 2.600, Avrg. sim: 0.193

Best sequence: 10
Seq.: 9, lev: 1 sim: 0.903, data: 1 2 3 4 5 7 8 7 8 7 8 7 8
                        7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 10
Seq.: 18, lev: 8 sim: 0.361, data: 2 3 4 5 7 8 7 8 7 8 7 8 7
                                    8 7 8 7 8 7 8 12 10
Seq.: 10, lev: 0 sim: 1.000, data: 2 3 4 5 7 8 7 8 7 8 7 8 7
                          8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 10
Avrg. Levenshtein Distance: 4.500, Avrg. sim: 0.211

Best sequence: 16
Seq.: 16, lev: 0 sim: 1.000, data: 1 2 3 4 5 7 8 12 10
Seq.: 13, lev: 1 sim: 0.702, data: 2 3 4 5 7 8 12 10
Avrg. Levenshtein Distance: 1.000, Avrg. sim: 0.234

Best sequence: 12
Seq.: 12, lev: 0 sim: 1.000, data: 1 2 9 10
Seq.: 11, lev: 1 sim: 0.422, data: 2 9 10
Avrg. Levenshtein Distance: 1.000, Avrg. sim: 0.422
```

■

Another output of the experiment is set of the new clusters C_ρ and the information of sequences contained within these clusters. The example of graph with clusters (behavioural patterns) with similar sequences of events performed by agents in the multi-agent system from Example 4.1, are shown in Figure 2.

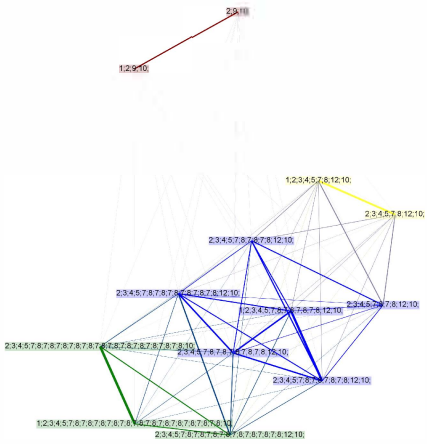

Figure 2. Visualization of Behavioural Patterns

The example of reduced user (agent) profiles created using behavioural patterns (sequence representatives) and their comparison with base user (agent) profiles is presented in Table II.

V. CONCLUSION

Systems, which allow the users the variability in their behaviour (systems without a given structure or with unrestricted rules for the user behaviour) record log files, from which is usually obtained a large amount of sequences, often very similar. We propose a new approach which allows us

Agent ID	Base Agent Profiles			Reduced Agent Profiles		
	Seq. ID	Count	Sequence Description	Seq. ID	Count	Sequence Description
sa0017	0	1	0	5	1	0
	1	243	6 6 6 6 6 6 6 6 6 6 6 6	6	243	6 6 6 6 6 6 6 6 6 6 6 6
	5	1	6 6 6 6 6 6 11	0	4	6 6 6 6 6 11
	3	8	6 11	7	8	6 11
	4	3	6 6 11			
ca0175	12	1	1 2 9 10	4	29	2 9 10
	10	21	2 3 4 5 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 10	3	21	1 2 3 4 5 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 10
	11	28	2 9 10	2	2	2 3 4 5 7 8 7 8 12 10
	15	1	2 3 4 5 7 8 7 8 12 10			
	13	1	2 3 4 5 7 8 12 10			

Table II
EXAMPLE OF BASED AND REDUCED AGENT PROFILES

the clear visualization of the latent ties between the users (or groups of users) with similar behaviour in such systems. The construction of a clear user network is not possible using common methods in these cases due to the large dimension of the base user profile vector. It was one of the reasons for finding the user behavioural patterns. The second reason was to clearly describe the system from the global point of view by finding the representative types of user behaviour, behavioural patterns.

ACKNOWLEDGMENT

This work was partially supported by the grant of Silesian University in Opava, Czech Republic, No. SGS/06/2013 'Advanced Modeling and Simulation of Economic Systems'.

REFERENCES

[1] R. Šperka, "Agent-based design of business intelligence system architecture," *Journal of Applied Economic Science*, vol. VII, no. 3(21), pp. 326–333, 2012.

[2] K. Slaninová, J. Martinovič, P. Dráždilová, D. Vymětal, and R. Šperka, "Analysis of agents' behavior in multiagent system," in *24th European Modeling and Simulation Symposium, EMSS 2012*, 2012, pp. 169–175. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84871505092&partnerID=40&md5=7b3f474a10d959a5604cf610548c37cd

[3] P. Dráždilová, J. Martinovič, and K. Slaninová, "Spectral clustering: Left-right-oscillate algorithm for detecting communities," *Advances in Intelligent Systems and Computing*, vol. 185, pp. 285–294, 2013.

[4] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters, "Workflow mining: a survey of issues and approaches," *Data Knowl. Eng.*, vol. 47, no. 2, pp. 237–267, 2003.

[5] W. M. P. van der Aalst, H. A. Reijers, and M. Song, "Discovering social networks from event logs," *Comput. Supported Coop. Work*, vol. 14, no. 6, pp. 549–593, 2005.

[6] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, 1st ed. Springer Heidelberg, 2011.

[7] M. E. J. Newman, *Networks: An Introduction.* Osford University Press, 2010.

[8] K. Musiałł and P. Kazienko, "Social networks on the internet," *World Wide Web*, vol. 1, pp. 1–42, 2012.

[9] K. Saeed, R. Chaki, A. Cortesi, and S. Wierzcho, Eds., *Extraction of Agent Groups with Similar Behaviour Based on Agent Profiles*, ser. Lecture Notes in Computer Science, vol. 8104. Springer Berlin Heidelberg, 2013. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40925-7_32

[10] D. Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 2008.

[11] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *J. ACM*, vol. 24, pp. 664–675, October 1977. [Online]. Available: http://doi.acm.org/10.1145/322033.322044

[12] A. Guo and H. Siegelmann, "Time-warped longest common subsequence algorithm for music retrieval," in *Proceedings of 5th International Conference on Music Information Retrieval ISMIR 2004*, C. L. Buyoli and R. Loureiro, Eds. Universitat Pompeu Fabra, 2004, pp. 258–261. [Online]. Available: http://ismir2004.ismir.net/proceedings/p049-page-258-paper113.pdf

[13] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *J. ACM*, vol. 51, no. 3, pp. 497–515, May 2004.

[14] D. Cheng, R. Kannan, S. Vempala, and G. Wang, "On a recursive spectral algorithm for clustering from pairwise similarities," MIT, Tech. Rep., 2003.

[15] J. Martinovič, P. Dráždilová, K. Slaninová, T. Kocyan, and V. Snášel, "Left-right oscillate algorithm for community detection used in e-learning system," in *CISIM 12. IEEE Computer Society.*, 2012.