Adv. Radio Sci., 9, 187–193, 2011
www.adv-radio-sci.net/9/187/2011/
doi:10.5194/ars-9-187-2011

Advances in Radio Science

Context-based user grouping for multi-casting in heterogeneous radio networks

C. Mannweiler, A. Klein, J. Schneider, and H. D. Schotten

Chair for Wireless Communications and Navigation, University of Kaiserslautern, Germany

Abstract. Along with the rise of sophisticated smartphones and smart spaces, the availability of both static and dynamic context information has steadily been increasing in recent years. Due to the popularity of social networks, these data are complemented by profile information about individual users. Making use of this information by classifying users in wireless networks enables targeted content and advertisement delivery as well as optimizing network resources, in particular bandwidth utilization, by facilitating group-based multi-casting. In this paper, we present the design and implementation of a web service for advanced user classification based on user, network, and environmental context information. The service employs simple and advanced clustering algorithms for forming classes of users. Available service functionalities include group formation, context-aware adaptation, and deletion as well as the exposure of group characteristics. Moreover, the results of a performance evaluation, where the service has been integrated in a simulator modeling user behavior in heterogeneous wireless systems, are presented.

1 Introduction

Group communication is of particular interest in wireless (access) networks, one example being the domain of multi-casting, i.e. simultaneously delivering the same content to multiple recipients. Chalmers and Almeroth (2001), Janic and Van Mieghem (2009), as well as Baumung and Zitterbart (2009), among others, have shown that multi-casting technologies can considerably contribute to more efficiently exploiting available network resources when applied under appropriate circumstances. A typical use case for exploiting multi-casting is a large-scale sport event with tens of thousands of spectators that can be grouped according to their user profile and context. Based on that classification, different groups can receive adapted content. However, a user classification service, as required in this use case, can also be exploited for selecting people with particular characteristics. Consumers with an affinity to a certain product category could be identified for specific marketing purposes. Both use cases would require a reliable means of data privacy and security as well as the user's approval.

The remainder of this paper is organized as follows: Sect. 2 presents the algorithms used for our user classification service and Sect. 3 briefly discusses implementation aspects. Section 4 evaluates the algorithms according to a comprehensive set of criteria. Section 5 concludes the paper with a short summary and outlook.

Correspondence to: C. Mannweiler (mannweiler@eit.uni-kl.de)

2 Classification methods

In this section, we will briefly introduce the clustering methods used for a context-aware user classification service. This includes the algorithm of the respective method as well as relevant characteristics such as hard vs. soft mapping or variable vs. fixed number of clusters. The notation employed in this paper is defined below:

- $\boldsymbol{C}$: Set of all clusters, $\boldsymbol{C} = \{\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_k\}$
- $\boldsymbol{N}$: Set of all cluster centers (nodes), $\boldsymbol{N} = \{N_1, N_2, ..., N_k\}$
- $\boldsymbol{K}$: Context space, $\boldsymbol{K} \subseteq \mathbb{R}^l$
- $\boldsymbol{X}$: Set of observations (user's context) to classify, $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}, \boldsymbol{x}_j \in \boldsymbol{K}$
- n: Number of observations (users) to classify
- k: Number of clusters (nodes)
- l: Dimension of context space
- $\boldsymbol{\Omega}$: Tuple of a cluster $\boldsymbol{C}_i$ and an observation $\boldsymbol{x}_j$, $\boldsymbol{\Omega} = (\boldsymbol{C}_i, \boldsymbol{x}_j)$
- $\boldsymbol{\mu}_i$: Center of cluster $\boldsymbol{C}_i$, $\boldsymbol{\mu}_i \in \mathbb{R}^l$
- $\boldsymbol{w}_i$: Weighing vector of node N_i (cluster center), $\boldsymbol{w}_i \in \mathbb{R}^l$
- t: Iteration counter

2.1 Methods with fixed number of cluster centers

Common to the methods presented in this subsection is the requirement to set the number of clusters before the start of the classification algorithm. During runtime, the number of clusters does not change.

2.1.1 K-Means algorithm

K-Means (MacQueen, 1967) is an algorithm that establishes a hard mapping between $\boldsymbol{C}_i$ and an observation $\boldsymbol{x}_j$, i.e. an observation is unambiguously associated with one cluster. Across multiple iterations, the following error function is minimized:

$$E = \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in \boldsymbol{C}_i} ||\boldsymbol{x}_j - \boldsymbol{\mu}_i||^2 \quad (1)$$

The iteration steps of the algorithm are:

1. *(Random) initialization of k cluster centers (nodes)* – cluster centers are randomly initialized in the l-dimensional context space.

2. *Mapping of observations to centers* – each observation is mapped to the closest node based on the selected distance metric such as Euclidean distance.

3. *Update of cluster centers* – position of nodes is recomputed based on the observations that are assigned to the node. Go back to step 2.

The algorithm terminates as soon as a specified termination criterion is reached, e.g. the error function falls below a threshold. For further details on K-Means, the reader is referred to Steinhaus (1957).

2.1.2 Neural Gas

The Neural Gas algorithms establishes a graph of k nodes that, independently of each other, move through the context space during the iterations. The core idea is to present available observations to the graph and to accordingly adjust the cluster centers. In brief, the steps of the algorithms are as follows:

1. *(Random) initialization of k cluster centers (nodes)* – Cluster centers are randomly initialized in the l-dimensional context space.

2. *Presentation of an observation* – An observation $\boldsymbol{x}_c \in \boldsymbol{X}$ is randomly chosen and presented to the graph. (An observation can be drawn multiple times.)

3. *Sorting of nodes* – Nodes are sorted according to their Euclidean distance to $\boldsymbol{x}_c$.

4. *Adjustment of node positions* – The position $\boldsymbol{w}_i$ of the graph's nodes within the context space is adjusted according to the following equation:

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + l(t) \cdot h_d(i) \cdot (\boldsymbol{x}_c - \boldsymbol{w}_i(t)) \quad (2)$$

where $l(t)$ is the learning rate of the current iteration ($l(t) > l(t-1)$ for all t) and $h_\mathrm{d}(i)$ the adjustment for an individual node depending on its ranking in step 3. If the termination criterion (e.g. number of iterations) has not been reached yet, the next iteration can begin, starting with step 2. Otherwise, observations are assigned to the closest node.

For further details on the algorithm, the reader is referred to Martinez and Schulten (1991) and Fritzke (1997).

2.1.3 K-Fixed-Means

None of the presented algorithms considers fixed cluster centers. However, for some of the use cases described in Sect. 1, this is an important alternative for user classification. Therefore, we have developed the so-called K-Fixed-Means algorithm, where the position of a cluster center is fix except for a defined tolerance interval (for each dimension of the context space) within which the cluster center can move. Moreover, the maximum distance to a center is limited, i.e. some observations may not be assigned to any node at all. Figure 1 depicts the idea of tolerance space (purple rectangle) and maximum distance (blue circle). The basic steps of the algorithm are:

1. *Initialization* – k cluster centers (nodes) are initialized at the determined (and fix) positions.

2. *Mapping of observations to centers* – each observation is mapped to the closest node based on the selected distance metric such as Euclidean distance. Non-assigned observations (i.e. those whose distance is above the defined thresholds) are collected in a set $\boldsymbol{U}$.

3. *Adjustment of cluster centers* – for each observation in $\boldsymbol{U}$, it is checked whether there exists a node that, if moved within its tolerance room, can accommodate the given observation. If their exists such a node, it is moved according the following equation:

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + l(t) \cdot d(\boldsymbol{x}_c, \boldsymbol{w}_i) \cdot (\boldsymbol{x}_c - \boldsymbol{w}_i(t)) \quad (3)$$

where $l(t)$ is the learning rate and $d(\boldsymbol{x}_\mathrm{c}, \boldsymbol{w}_i)$ a factor that takes into account the different tolerance intervals. After all observations in $\boldsymbol{U}$ have been checked, the next iteration starts with step 2, unless the criterion for termination is met.

As for the Neural Gas algorithm, the learning rate decreases in later iterations.

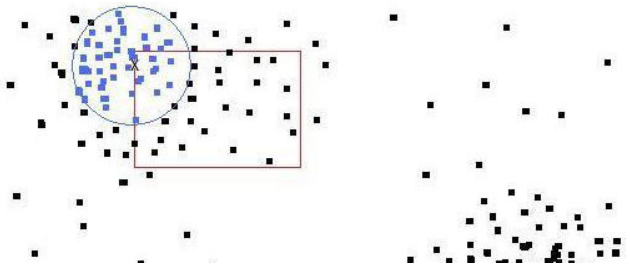

Fig. 1. Concept of K-Fixed-Means algorithm.

2.1.4 Fuzzy-C-Means

In contrast to the algorithms presented so far, Fuzzy-C-Means can assign an observation to several clusters. The degree u_{ij} of membership of an observation $\boldsymbol{x}_j$ to a single cluster $\boldsymbol{C}_i$ has to lie within the interval $]0,1]$ and the sum of all memberships of an observation must add up to 1, i.e. $\sum_{i=1}^{k} u_{ij} = 1$ for any j. The algorithm minimizes the following error function:

$$E = \sum_{i=1}^{k}\sum_{j=1}^{n} u_{ij}^{m} ||\boldsymbol{x}_j - \boldsymbol{\mu}_i||^2, 1 \leq m \leq \infty \tag{4}$$

The exponent m is the so-called "fuzzifier". The higher its value, the fuzzier the mappings of observations to nodes. In practice, values between 1 and 2.5 have proven to generate good clustering results (Bezdek, 1981). The basic steps of the algorithm are:

1. *Initialization* – (random) initialization of values u_{ij}
2. *(Re)calculation of cluster centers* – for the current iteration step, the cluster centers are calculated according to

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} \boldsymbol{x}_j}{\sum_{j=1}^{n} u_{ij}^{m}} \tag{5}$$

3. *Recalculation of degrees u_{ij} of membership* – the degree of membership of an observation $\boldsymbol{x}_j$ to a cluster $\boldsymbol{C}_i$ is calculated according to the following equation:

$$u_{ij} = \frac{1}{\sum_{l=1}^{k} \frac{||\boldsymbol{x}_j - \boldsymbol{\mu}_i||^{\frac{2}{m-1}}}{||\boldsymbol{x}_j - \boldsymbol{\mu}_l||}} \tag{6}$$

4. *Test of termination criterion* – if the termination criterion, e.g. sum of changes of u_{ij} for all combinations i, j is below a defined threshold, is not satisfied yet, another iteration is performed starting with step 2.

The presented algorithm converges to a local minimum which is not necessarily the optimal solution. Moreover, the results depend on the initialization of u_{ij}.

2.2 Methods with variable number of cluster centers

In contrast to the algorithms presented so far, this class of algorithms is capable of adjusting the number of cluster centers during runtime. Hence, not only the assignment of observations to clusters but also the number of clusters is optimized. Using the algorithms from the previous section, this could only be achieved by several runs with different amounts of clusters.

2.2.1 Growing Neural Gas

The Growing Neural Gas algorithm is an extended version of the Neural Gas algorithm as presented in the previous section. By the insertion and aging of edges between cluster centers (nodes) according to a set of rules, the topology of the underlying data shall be reflected more precisely. For a detailed description of the algorithm, the reader is referred to Fritzke (1995). Here, only a short overview of the algorithm steps is presented to sketch the basic idea:

1. *Initialization* – two nodes are randomly put in the context space (without an edge between them).
2. *Selection of observation and calculation of closest nodes* – from the set of observations, one is picked (randomly) and the closest node (N_1) as well as the second closest (N_2) are determined based on the Euclidean distance
3. *Insertion of edges* – in case there is no edge between N_1 and N_2, it is inserted. In any case, the age of the edge is set to 0.
4. *Calculation of a node's statistical error value* – for every node, the statistical error E_{N_i} is stored. It represents the total error of all observations assigned to that node. For N_1 (as determined in step 2), the value is updated by adding its Euclidean distance to the current observation $\boldsymbol{x}_c$:

$$E_{N_1}(t) = E_{N_1}(t-1) + ||\boldsymbol{w}_{N_1} - \boldsymbol{x}_c||. \tag{7}$$

5. *Adaptation of node positions* – the position of N_1 as well as its direct topological neighbors is adapted as follows:

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + l \cdot (\boldsymbol{x}_c - \boldsymbol{w}_i(t)). \tag{8}$$

   The learning rate l, though being constant during all iterations, is different for N_1 and its neighbors, i.e. for the neighbors, it is approximately two orders of magnitude lower than for N_1.

6. *Aging of edges* – for all edges originating from N_1, the age is incremented by 1. If the age has passed a defined threshold, the edge is removed. If this results in a node without any edges, it is removed as well.

	Allows for Variable Number of Clusters	Allows for Fuzzy Clustering	Allows for Clustering of Nominal Data	Allows for Pre-defined Cluster Centers
K-Means	No	No	Yes	No
Neural Gas	No	No	No	No
Growing Neural Gas	Yes	No	No	No
Fuzzy-C-Means	No	Yes	Yes	No
K-Fixed-Means	No	No	No	Yes

Fig. 2. Overview of algorithm characteristics.

7. *Insertion of a new node* – after a defined number of iterations has passed, the node $N_{E_{\max}}$ with the highest accumulated error $E_{\max}$ is determined. Among its direct topological neighbors, the one with the highest accumulated error $E_{n_{\max}}$ is chosen and a new node is placed midway the selected nodes. The edge between $N_{E_{\max}}$ and $N_{E_{n,\max}}$ is removed and edges between the new node and $N_{E_{\max}}$ and $N_{E_{n,\max}}$, respectively, are inserted. The accumulated error of the old nodes is decreased by a factor α. The accumulated error of the new node is set to arithmetic mean of these two (updated) values.

8. *Reduction of accumulated error* – for all nodes, the accumulated error is decreased by a factor $\beta \ll \alpha$.

9. *Test of termination criterion* – if the termination criterion (e.g. current number of nodes) is not fulfilled, the next iteration starts at step 2.

A summarizing overview of the employed algorithms and their characteristics is given in Fig. 2.

3 Implementation

Implementation of the user classification service has been guided by several objectives, most importantly:

– high availability of the service

– high scalability

– simple extensibility

– usage of widely adopted protocols and data representation formats

– adequate latency behavior

Taking these aspects into account, we designed a service creation and delivery environment utilizing the JavaEE platform in conjunction with the JBOSS Application Server (version 5.1.0.GA). Service functionality can be reached via http requests and, optionally, additional transmission of XML data. The service hence implements a so-called RESTful (Tyagi, 2006) interface and can be made available to basically any hardware/software platform and (almost) independently of the kind of access network. Available functionality currently includes group formation, context-aware group adaptation, and deletion as well as the provisioning of characteristics of active groups.

4 Results and performance evaluation

For evaluating the performance of the presented algorithms, the following set of criteria has been defined:

– Quality of clustering results – quality is given by the total classification error of a given result.

– Temporal performance – this criteria measures the total time necessary for a classification.

– Stability – evaluates similarity of outcomes for multiple runs with identical data set.

– Flexibility – are algorithms capable of handling other than metric data?

– Implementation – evaluates the amount of effort necessary for algorithm implementation

4.1 Quality and temporal performance of clustering methods

For a comprehensive evaluation within the given multi-casting scenario, every algorithm has been executed 100 times for any given input parameter combination. Input parameters included number of entities (840 and 8400), number of groups to be formed (4, 6, 9, 13, 19, 28), and dimension of context space (2 and 5). We analyzed total execution time of an algorithm, its variance as well as total accumulated error. In summary, most important observations and results are (a more detailed performance analysis of the individual algorithms can be found in Appendix A):

4.1.1 K-Means

K-Means is among the best performing algorithms. However, computation times significantly increase with higher

	Stability	Flexibility	Implementation Effort
K-Means	High	Very high	Low
Neural Gas	High	Medium	Medium
Growing Neural Gas	High	Medium	High
Fuzzy-C-Means	Very High	High	Low
K-Fixed-Means	Deterministic	Low	Medium

Fig. 3. Summary evaluation of classification algorithms.

number of users and groups. In terms of the absolute classification error, it achieves the best result of all analyzed algorithms with an average of 133.52 across 100 runs of the standard scenario.

4.1.2 Neural Gas

Similar to K-Means, Neural Gas is a relatively fast algorithm. Moreover, runtime decreases significantly for larger amount of groups because assignment of observations, in contrast to K-Means, is done only once and not in every iteration. Moreover, with higher numbers of nodes, their final position is reached much faster. The average classification error was 133.54.

4.1.3 Growing Neural Gas

In terms of both temporal behavior and dependency on entity as well as group count, the algorithm behaves very similar to Neural Gas. Moreover, its average classification error was only slightly higher, averaging at 137.74.

4.1.4 Fuzzy-C-Means

With regard to performance, the algorithm performs worse with increasing numbers of observations, variables per observation and number of groups. Due to its high computational effort, the algorithm cannot handle more than ten groups in a timely manner. For the calculation of the average classification error (134.69), observations have been assigned to the node they have had the highest affiliation to.

4.1.5 K-Fixed-Means

Overall, algorithm runtime is on a satisfying level and comparable to that of K-Means. For five variables (i.e. a five-dimensional context space), the algorithm arrives in a stable state earlier since (with the given set of test observations) the cluster centers reach their final position faster. A comparison of the accumulated classification error does not make sense since some observations were not classified at all.

4.2 Overall evaluation

A brief summary of the remaining evaluation criteria is depicted in Fig. 3.

Overall, the K-Means algorithm disposes of the most appropriate characteristics in the given multi-casting use case. Not only good results in terms of quality and performance but also its capability to handle nominal data makes it the default choice for classification requests. Neural Gas is the preferred algorithm for large entity and group counts; however, it should not be used for clustering observations with nominal data. In case that the number of groups is not known yet, Growing Neural Gas, despite its difficult parametrization, is the best choice. Fuzzy-C-Means is especially recommended if the number of groups remains small and the structure of the observation set is rather complex. Finally, the K-Fixed-Means algorithm should be chosen if cluster centers, i.e. group characteristics, are pre-determined and should only be marginally changed during execution.

5 Conclusions

This paper has presented an evaluation of different user classification algorithms for enabling group communication and multi-casting in wireless networks. Classification tests have been performed based on simulated context information about users (such as location, music taste, and age) with different numbers of both users and expected groups. K-Means and (Growing) Neural Gas as well as the newly developed K-Fixed-Means have consistently recognized the basic structure within the set of users and produced fast and stable classification results with low total errors.

A major aspect of future work is the development of classification methods that can handle nominal (and ordinal) data since most of the interesting user data enabling multi-casting (such as a user's profile) fall into these categories. Moreover, the implemented service will be verified in a testbed environment where efficiency gains based on the realization of multi-casting will be analyzed quantitatively.

Appendix A

Algorithms performance results

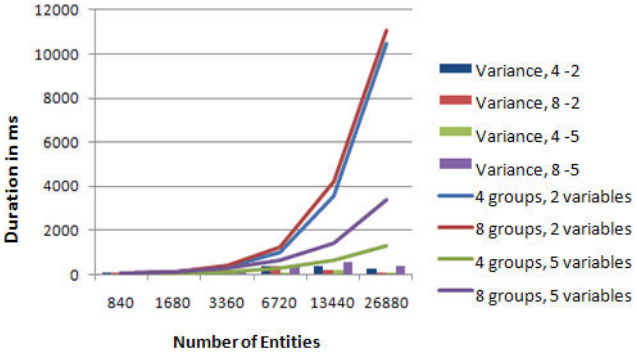

Fig. A1. Performance of K-Fixed-Means algorithm.

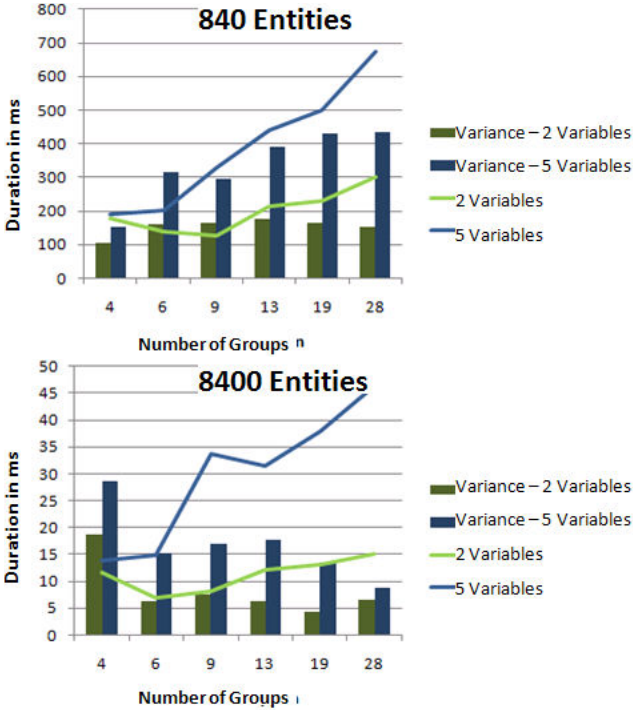

Fig. A2. Performance of K-Means algorithm.

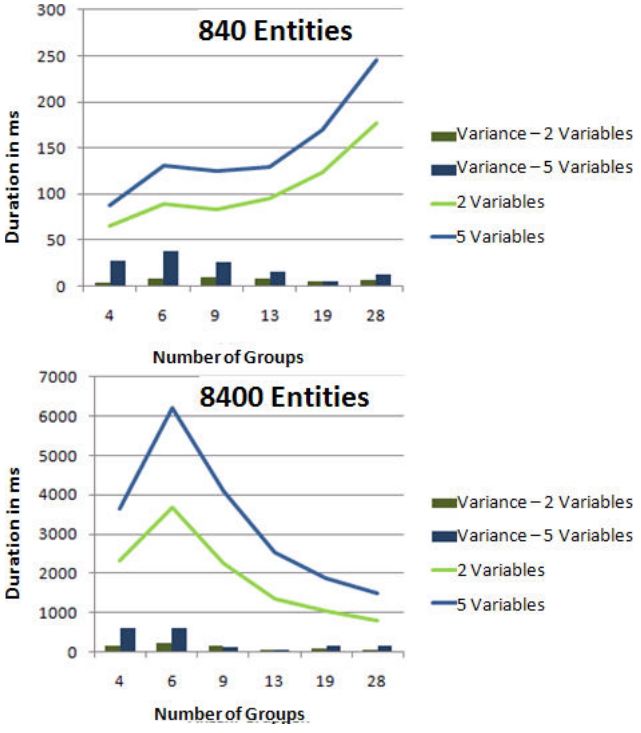

Fig. A3. Performance of Neural Gas algorithm.

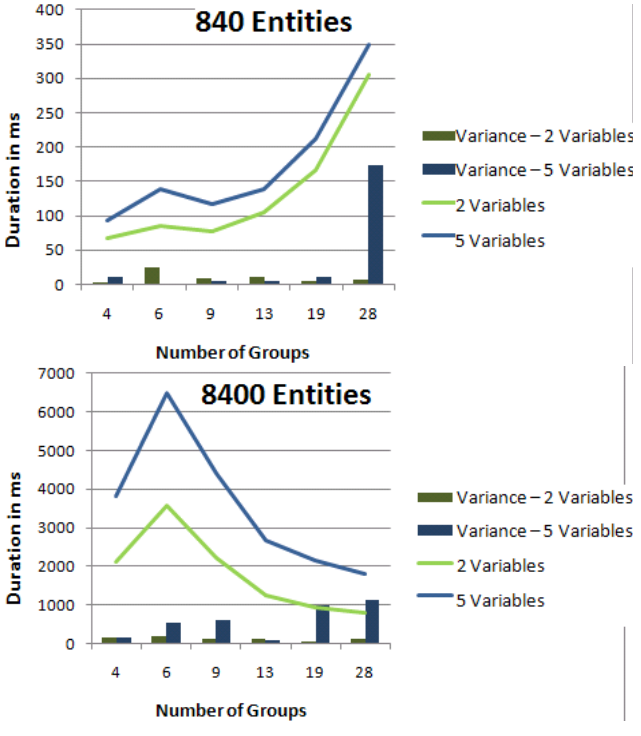

Fig. A4. Performance of Growing Neural Gas algorithm.

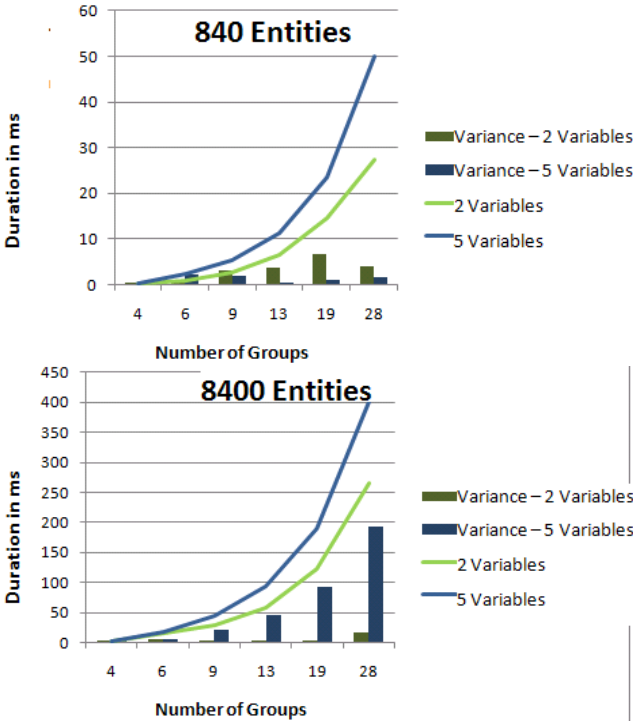

Fig. A5. Performance of Fuzzy-C-Means algorithm.

Acknowledgements. Parts of this work were funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Foerderkennzeichen 01 BK 0808, GLab). The authors alone are responsible for the content of the paper.

References

Baumung, P. and Zitterbart, M.: MAMAS – Mobility-aware Multicast for Ad-hoc Groups in Self-organizing Networks, in: Basissoftware für drahtlose Ad-hoc- und Sensornetze, edited by: Zitterbart, M. and Baumung, P., pp. 33–48, Universitaetsverlag Karlsruhe, March 2009.

Bezdek, J. C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York, 1981.

Chalmers, R. and Almeroth, K.: Modeling the Branching Characteristics and Efficiency Gains of Global Multicast Trees, Proceedings of IEEE Infocom, Anchorage (AK), 2001.

Fritzke, B.: A growing neural gas network learns topologies, in: Advances in Neural Information Processing Systems 7, edited by: Tesauro, G., Tesauro, G., Touretzky, D. S., and Leenet, T. K., MIT Press, Cambridge (MA), pp. 625–632, 1995.

Fritzke, B.: Some Competitive Learning Methods. Java Paper, 1997, http://www.neuroinformatik.ruhr-uni-bochum.de/VDM/research/gsn/JavaPaper/, accessed 11 November 2010.

Janic, M. and Van Mieghem, P.: The Gain and Cost of Multicast Routing Trees. International Conference on Systems, Man and Cybernetics (IEEE SMC 2004), The Hague, 2004.

MacQueen, J.: Some methods for Classification and Analysis of Multivariate Observations: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297, 1967.

Martinez, T. M. and Schulten, K. J.: A neural gas network learns topologies, in: Artificial Neural Networks, edited by: Kohonen, T., Mäkisara, K., Simula, O., Kangas, J. Elsevier, North-Holland, pp. 397–402, 1991.

Steinhaus, H.: Sur la division des corps matiriels en parties, Bull. Acad. Polon. Sci., 4(12), 801–804, 1956.

Tyagi, S.: RESTful Web Services. Oracle Technology Network, 2006, http://www.java.sun.com/developer/technicalArticles/WebServices/restful/, accessed 28 November 2010.