Basic Clustering Algorithms Used for Monitoring the Processes of the ATM's OS

Michal Maliszewski
University of Silesia,
Faculty of Computer and Materials Science
41-200 Sosnowiec, Poland
Email: mmaliszewski@us.edu.pl

Urszula Boryczka
University of Silesia,
Faculty of Computer and Materials Science
41-200 Sosnowiec, Poland
Email: urszula.boryczka@us.edu.pl

***Abstract*—The number of highly sophisticated software-based attacks on ATMs is growing these days. New types of threats require new ways of system protection. Most secure solutions are based on whitelists and sandboxes, which unlike antivirus solutions are able to protect the system against any new threat. Sadly, they are hard to configure therefore require an expert-level knowledge of operating system mechanisms and software security techniques. The main purpose of this article is to present the possibilities of using clustering algorithms in a configuration process of a sandbox-based security solution. Results of the experimental studies show that even basic clustering algorithms can be used as a part of the configuration process. Achieved results were deemed promising by the control algorithm and thus can be used as a base for future research.**

***Index Terms*—Data clustering, security, sandbox, whitelisting, operating system.**

I. Introduction

Automated teller machine (ATM) is an electronic telecommunications device that enables the customers of a financial institution to perform financial transactions, particularly cash withdrawal, without the need for a human cashier, clerk or bank teller. According to the newest International Monetary Fund's report, there are now close to 3 million ATMs installed worldwide [1].

Sandbox (computer security) is a security mechanism for separating running programs, however it is usually used to execute untrusted programs or code without risking harm to the operating system [2].

Application whitelisting is a security mechanism used to prevent unauthorized programs from running. The goal of whitlisting is primarily to protect computers and networks from harmful applications and to a lesser extent, to prevent unnecessary demand for resources. Whitelisting is the opposite of blacklisting, a method used by most antivirus programs, intrusion detection/prevention systems and spam filters [3].

Sandboxes allow to control the processes flow within the operating system. The main idea of this kind of solution is to create a limited number of virtual sandboxes and then control access to their sources. Everything that is not defined as an exception is forbidden.

It is, of course, possible to simplify the configuration process, however grouping and configuring the processes for the first time can be a major problem. The most common issues include:

- What kind of processes should be included?
- How to group processes into sandboxes?
- How many groups should be created?

One of the security software solution based on whitelisting is software provided by American company - Symantec. Their solution allows for monitoring flow of the processes running on operation system. Program called Symantec Critical System Protection (SCSP) is a security wrapper for mission-critical environments. This software has the ability to protect critical systems such as Scada, ATMs and point-of-sale terminals [4].

In most cases the first configuration is done by the Bank's Security Department. The configuration is done manually by the employee who tries to find the most secure configuration build. The employee has to discover all necessary programs on the ATM and find out what kind of resources they use (e.g. files, registries or network sources). Once this is done, the configuration is then put on the ATM and is monitored and improvements are added in case of any security or hardware issues. The configuration can be applied to the ATM using e.g. SCSP console or different security solution like Diebold Nixdorf Terminal Security-Intrusion Protection [5]. As of this moment there has not been any propositions to use different, than manual, methods of configuration.

In literature there is a view that whitelisting is the future of software security. That point was represented over the last few years, for example in *The Promise of whitelisting* by Steve Mansfield-Devin [6]. Some researchers are focusing on the areas where whitelisting can be used, for example C. Bildsten in *Application Whitelisting: Smartphones in High Security Environments* [7] or R.Barbosa, R.Sadre, A.Pras in *Flow whitelisting in SCADA networks*, where authors propose a way to improve the security of SCADA networks using flow whitelisting [8]. On the other hand, clustering algorithms are widely use in intrusion detection systems (a blacklisting approach), e.g. cyber attacks detection in PCS systems described in *A clustering-based approach to detect cyber attacks in process control systems* by I. Kiss, B. Genge and P. Haller [9].

The most similar approach was presented in 2016 by S.I. Monterrosa Reyes, G. R. Salgado and J. P. Ortega in

978-1-5090-5795-5/17/$31.00

Defining Adaptive Whitelists by Using Clustering Techniques, a Security Application to Prevent Toll Fraud in VoIP Networks, where authors used *K*-means to generate whitelists with call destinations [10], however this method was used in case of VoIP Networks. Moreover, proposed whitelisting mechanism is adaptive, which will be difficult to use in case of ATM security (e.g. SCSP will not allow newly installed program to run without updating the policy, what must be somehow confirmed).

The idea presented in this article is to use clustering algorithms to prepare the initial SCSP configuration and it is proposed due to a lack of satisfactory results achieved by simple classifiers in earlier research done by the authors. In previous approach author focused more on the processes properties instead of connections between them. Such concept required also a huge database of templates (as a training set for classifiers) which should be continuously updated.

Methods used within the article are not the only possible solutions, but their results can be used as a base for future optimization or clustering process improvement. One or both of the above will be necessary in case of clustering inaccuracies (in case of post-processing solving e.g. Minimum *K*-cut Problem may be the proper solution to reduce a redundant number of connections between groups) and additional constraints within clusters.

Minimum *K*-cut Problem is, in mathematics, a combinatorial optimization problem that requires finding a set of edges, which when removed would partition the graph to k connected components. These edges are referred to as *k*-cut. The goal is to find the minimum-weight k-cut. This partitioning can have applications in VLSI design, data-mining, finite elements and communication in parallel computing [11], [12].

II. Clusterization of Processes Based on SCSP Log Files

Symantec Critical System Protection controls and monitors what programs and users can do to computers. Agent software at the endpoints controls and monitors behavior based on a policy. The Symantec Critical System Protection policy library contains prevention and detection policies that can be used and customized to protect a network. Agents report events to the management console. Agent Log Rules control the events logged for a particular agent. Logged data includes event date and time, event type, importance rating, and any prevention action performed [13].

Data for this tests is taken from an ATM with Windows 7 SP 1 32-bit OS and includes two three-hour test runs with Terminal Security Intrusion Protection (which uses SCSP) installed in monitoring mode (every process is monitored but neither is yet blocked). That gives two datasets, each containing 2500 records. Each record is defined as a pair of the process and argument which is equal to process and its target source (like file or registry) with additional attributes. Number of records in dataset may seem small, but in fact, number of applications which are continuously running on ATM OS is also limited. Please, keep in mind that the author must have been able to prepare the correct configuration also manually, to evaluate the results obtained in relation to actual ATM security needs.

At first each dataset was filtered by event type. Only events categorized as requests for a file or registry were counted. Records generated by explorer.exe or system tools and services from /system32 were removed (as a part of many background OS operations they cause noise within the data itself).

SCSP log files contain a lot of useful information, however for the initial configuration only a handful truly matter. Duplicated records were removed (meaning the same Process Target pair was present). In result, the dataset contained unique Process-Target pairs.

Next step was to filter the dataset attributes. It is a complex process which requires a deep knowledge about Symantec Critical System Protection program as well as the structure of the log files. From the twenty-three available attributes only eight were useful for the configuration procedure. They were selected because of their importance in the whole configuration process. Tests were made in three variants: with two attributes (minimum contains ”Process” and ”Arguments” because this values are necessary to find connection between resources), with four attributes where additionally ”Event type” (necessary for preprocessing, e.g. filtering) and ”Disposition” (necessary for postprocessing, in case that SCSP was already configured on the ATM and data from that configuration can be used to improve current sandbox setup) and with eight attributes (additional informations which can be used to discover whether a particular process, for example want to write or read the target file - such informations are not given directly in log file). Below are selected attributes with short description:

- Event type (type of an event, e.g access to file or registry),
- Process (process, mostly a path to executable file),
- Disposition (action of an SCSP like allow or deny access),
- Arguments (this attribute is equal to the target source, saved as path to the source, called also as a ”target” later in this article),
- Message Type (type of a record: info, warn, err),
- Agent State,
- Event User (user who begin an action),
- Operation.

Minimum required attributes (2): Process, Arguments,

First choice attributes (4): Event Type, Process, Arguments, Disposition,

Maximum checked attributes (8): All above.

Processes and arguments were converted to the numerical values (unique integer values). Conversion was done by 1:1 mapping (each process/argument to integer values starting from 1, all occurrences of the same process has the same integer value). Using numbers, on a dataset so small, allows for a fast analysis result, which is much more difficult when comparing a full, absolute file or a registry path (in case of manually checking the results).

The result dataset is presented in the Figure 1 (Process-Target pairs without any connections to other sources were

not included in the graph. More complex dependencies were kept intact).

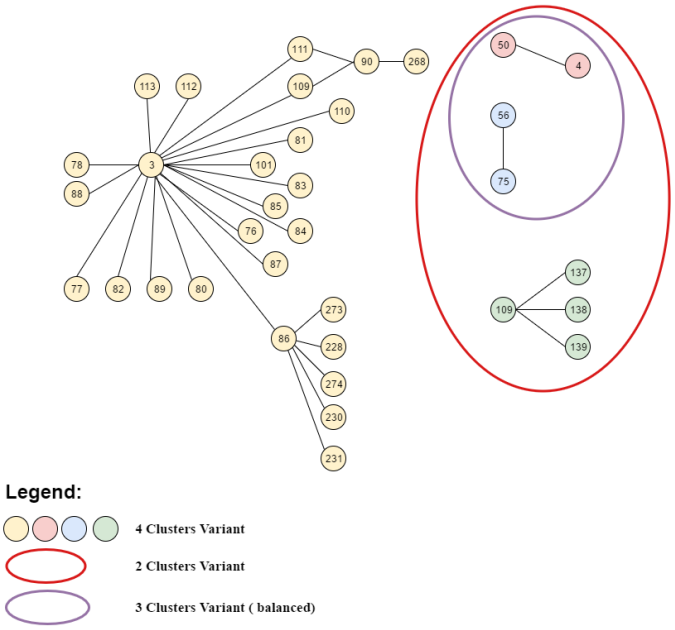

Fig. 1. Visualization of relations inside sample dataset

Selected clustering algorithms are listed below.

- *K*-means [14]
- *K*-medoids [15]
- *X*-means - is a variation of *K*-means clustering that refines cluster assignments by repeatedly attempting subdivision, and keeping the best resulting splits, until defined criterion is reached. The Bayesian Information Criteria is used to make the splitting decision. The KD-Tree is used to store the data and improve the algorithms speed [16].

Because it was author's first approach to process clustering in ATM environment *K*-means algorithm was a natural choice. However, the nature and complexity of the problem did not allow to set up clusters in advance, the dynamic approach to the problem had to be applied. For this reason, the *K*-means extended variant has been selected (*X*-means). As expected *K*-means variants have a low resistance against noises also in processes clustering, so in addition *K*-medoids algorithm have become a research subject.

Should there be a need to define the number of clusters from the beginning (*K*-means, *K*-medoids etc.) an optimal number has been chosen (basing on author experience with Terminal Security Intrusion Protection and knowledge about ATM security). The optimal number being 3 which is the most balanced clustering variant from all available solutions. For dynamic clustering, the minimum number of clusters was set to 2, because this is the minimum number of groups which can solve problem (see Figure 1). Max runs and optimization steps were set to 1000. Any of the used algorithms did not show an improvement with optimization steps or number of iterations higher than selected values.

Please note, that a non-dynamic clustering algorithms cannot be selected as a solution for this problem. In most cases there is no possibility to define the perfect number of sandboxes, that could be used, therefore dynamic clustering should be used. Static clustering methods such as *K*-means were used in comparison to dynamic method in order to estimate their efficiency.

III. Measuring Accuracy

It is possible to measure the test dataset manually, however due to high number of process-target pairs finding the perfect solution is impossible. There are, of course, several ways to check out clustering quality, but in this case, except measure, authors want to know exactly which process-target pairs were incorrectly clustered. Furthermore measures based only on element attributes may be incorrect, because we are investigating also connections between elements. For these reasons to measure results a special algorithm was implemented. It is counting number of the connections between clusters, choosing the most positive variant of given clustering result. The code of the algorithm itself can be found in Algorithm 1.

Algorithm 1: Algorithm to find quality of clustering operation

```
procedure countMeasure (file);
  Input : *.csv file with clustering results file
  Output: measure m, where m ∈ [0, 1]
sort file content by process;
c = count number of clusters;
rec = count total number of records in file;
foreach process p in file do
    find all p occurrences (as process or target) in file;
    cnum = count number of clusters contains process p;
    if cnum > 1 then
        take largest cluster as 'correct';
        err+ = count other clusters elements as
          'incorrect';
    end
    register process in 'checked' list chk;
end
foreach target t in file where t ∉ chk do
    repeat steps for processes;
end
if the minimum conditions are met correctness then
    m = ((rec − err)/rec);
end
else
    m = 0;
end
return m;
```

Where minimum conditions which met correctness means the limit of which the results are considered valid. In the simplest case it can check if number of clusters is greater than 1.

IV. Results

Results are presented as a number of misconfigured processes within sandboxes to a total number of records (sandbox mismatch). The lower the number, the better. Each method was checked with different type of measures (numerical or mixed). The results visible on the charts were created by comparison against ideal configurations done manually with specialists from Diebold Nixdorf Banking Security Division. Additionally, the results were compared against a validation algorithm (see algorithm 1).

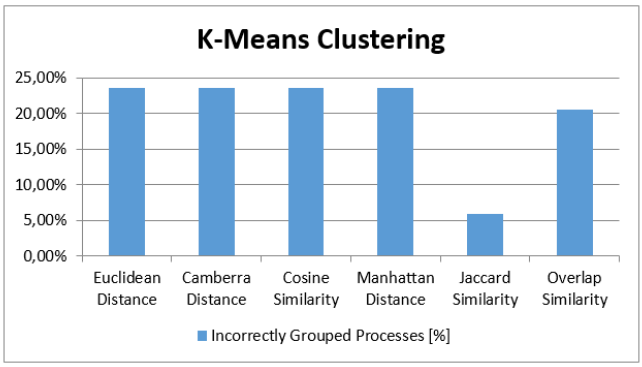

Fig. 2. *K*-means Clustering Algorithm Accuracy

With defined a priori number of clusters, *K*-means algorithm achieves good results, however the result of *K*-means with Jaccard Index was slightly better. It might seem to be a great solution, but further tests show that this type of *K*-means algorithm tends to put most of the elements together, which in more complex datasets is close to 1. The verification algorithm was able to detect such situations and set the measure to 0.0 (see table I).The chosen dataset does not have one perfect solution. After a manual configuration process author was able to provide three configurations with similar security level. An ideal number of clusters should be in range [2,4].

As mentioned before, *K*-medoids clustering is more resistant to noise. The results seem to be more stable and gives around 50% better outcome. Differences between measure types are smaller.

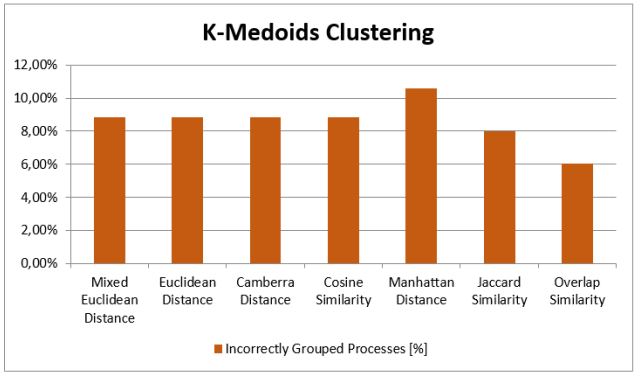

Fig. 3. *K*-medoids Clustering Accuracy

In addition we can observe that there is no difference between results based on numbers (processes converted to numerical values) and results achieved with Mixed Euclidean Distance (plain process paths). The reason for this is that both plain processes paths and their hash values has no connection between each others. Two programs monitored by SCSP which are in e.g C:\Program Files\... are not necessarily similar to each other (use different files, registry values, etc.).

After comparing the processes with the use of static clustering, it was time to use the *X*-means dynamic clustering. Results achieved with Euclidean and Manhattan distances were the same as in case of *K*-means clustering. Unfortunately, all others results were worse than their static counterparts. *X*-means with Jaccard Similarity achieved the best results, however the effect visible in the *K*-means algorithm was intensified. The algorithm was not able to group the data in more than two clusters. Increasing the minimum number of clusters did not change the effect.

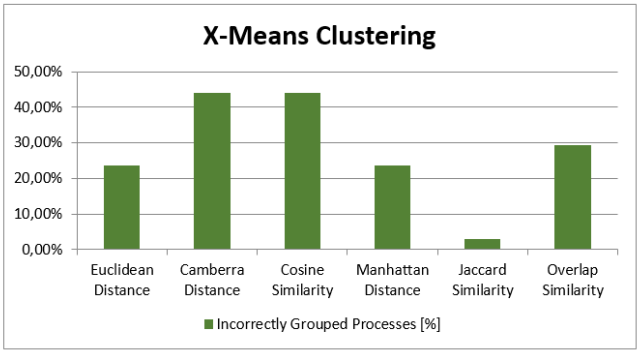

Fig. 4. *X*-means Algorithm accuracy

Comparison of all methods (both dynamic and static with the same measure types used) is presented in Figure 5.

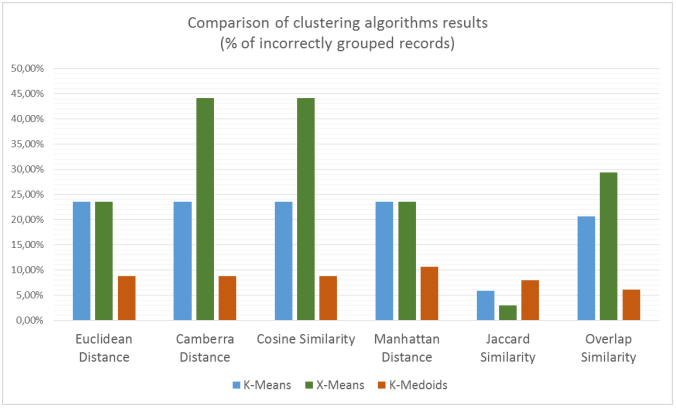

Fig. 5. Comparison of *K*-means, *K*-medoids and *X*-means Accuracy

Tabular form of the results validated against Algorithm 1 is presented in Table I.

Clustering with Jaccard and Overlap Similarities did not give good results against verification algorithm, what seems to confirm the theory about good results happening, because

TABLE I
COMPARISON OF K-MEANS, K-MEDOIDS AND X-MEANS ACCURACY

Measure Type/Algorithm	K-means	X-means	K-medoids
Euclidean Distance	0.8824	0.8286	0.8824
Camberra Distance	0.8824	0.7714	0.8824
Cosine Similarity	0.8824	0.7714	0.8824
Manhattan Distance	0.8824	0.8286	0.8824
Jaccard Similarity	0.9118	**0.0**	**0.0**
Overlap Similarity	0.9118	**0.0**	0.9412

of collecting elements in one group (in most cases algorithm gave a measure 0.0, in some cases value was close to 1 which means that minimum acceptance criteria was meet, but it was close to get 0.0 note).

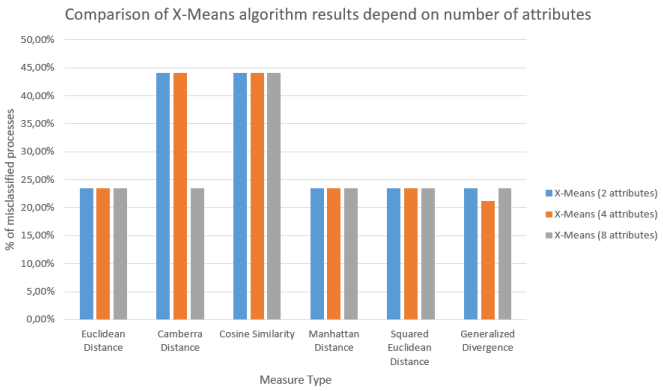

Fig. 6. Comparison of X-means algorithm results depend on number of attributes

V. CONCLUSIONS AND OTHER WORK

Static clustering algorithms, especially the K-medoids, can solve the clustering problem with an acceptable percentage of accuracy. Unfortunately they cannot be used in the process of sandbox grouping, due to a high number of elements and no possibility of checking whether the generated segmentation is correct. This kind of algorithms can be used to improve the accuracy of manual configuration, or as a user support feature, but in case of fully automated solution, the best idea would be to use dynamic clustering supported by postprocesses like solving the Minimum K-cut problem or to modify clustering algorithm to ensure that proper division occurs.

In best cases, the results of dynamic clustering were as good as their static counterparts. Misconfigured number of elements between 20% - 40% offers a great start for the future optimization algorithms. The way to go for this could be to find different clustering algorithms such as MajorClust [17], which is based on graph segmentation, and compare them with X-means results.

Evaluation of quality of the clustering process should be changed. A perfect configuration can be found for small datasets only, and cannot be verified on a large amount of data. Quality measures can be defined as a minimum number of edges between clusters in a correlation to number of groups (in order to avoid one cluster being counted as the perfect solution). Like it was shown in Algorithm 1.

MajorClust algorithm solution can be based on Graph Partition problem, in the same way as it is used currently for clustering and detection of cliques in social, pathological and biological networks [18]. It is an NP-Hard Problem, but with a maximum given number of elements in cluster, where maximum number of clusters should be possible to obtain for an optimal configuration. In case of graph partition for the processes, the problem can be the high number of clusters with not connected Process-Target pairs. This problem can be solved by balanced clustering or by merging clusters.

REFERENCES

[1] Automated teller machines (ATMs) (per 100,000 adults), International Monetary Fund, Financial Access Survey, "http://data.worldbank.org," [Online]. Available: http://data.worldbank.org/indicator/FB.ATM.TOTL.P5. [Accessed 28 04 2017].

[2] A secure environment for untrusted helper applications confining the Wily Hacker, I. Goldberg, D. Wagner, T. Randi i E. Brewer, in SSYM'96 Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography, San Jose, California, 1996.

[3] Application Whitelisting Gains Traction, J. Brooks, 25 09 2008. [Online]. Available: http://www.eweek.com/c/a/Security/Application-Whitelisting-Gains-Traction. [Accessed 25 09 2016].

[4] Symantec Critical System Protection, P. Stephenson, 01 03 2013. [Online]. Available: http://www.scmagazineuk.com/symantec-critical-system-protection/review/3842/. [Accessed 30 10 2016].

[5] Intrusion Protection Product Card, Diebold Nixdorf, "http://www.dieboldnixdorf.com," 1 8 2016. [Online]. Available: http://www.dieboldnixdorf.com/-/media/diebold/diebold-asset-library/software/en/intrusion_protection_productcard_aug2016_us.pdf. [Accessed 28 04 2017].

[6] The promise of whitelisting, S. Mansfield-Devine, Network Security Volume 2009, July 2009, Elsevier, pp. 46.

[7] Application Whitelisting: Smartphones in High Security Environments, C. Bildsten, Dissertation, 2013.

[8] Flow whitelisting in SCADA networks, R. R. R. Barbosa, R. Sadre, A. Pras, , International Journal of Critical Infrastructure Protection, Volume 6, Issues 34, December 2013, Pages 150-158, ISSN 1874-5482.

[9] A clustering-based approach to detect cyber attacks in process control systems, I. Kiss, B. Genge and P. Haller, 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), Cambridge, 2015, pp. 142-148.

[10] Defining Adaptive Whitelists by Using Clustering Techniques, a Security Application to Prevent Toll Fraud in VoIP Networks S,.I. Monterrosa Reyes, G. R. Salgado, J. P. Ortega, Proceedings of the International Conference on Information and Knowledge Engineering (IKE), Athens, 2016, pp. 100-106.

[11] Proc. 29th Ann. IEEE Symp. on Foundations of Comput. Sci., O. Goldschmidt and D. S. Hochbaum, White Plains, 1988, pp. 444-451.

[12] Computers and Intractability: A Guide to the Theory of NP-Completeness, M. R. Garey and D. S. Johnson, W.H. Freeman, 1979. ISBN 0-7167-1044-7.

[13] Symantec Critical System Protection Administration Guide, Symantec, Mountain View, 2007, pp. 16-18.

[14] Some Methods for classification and Analysis of Multivariate Observations. MacQueen, J. B. Berkeley : University of California Press, 1967. Vols. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1 Statistics, pp. 281-297.

[15] Clustering by means of Medoids, in Statistical Data Analysis Based on the L1Norm and Related Methods, Kaufman, L. and Rousseeuw, P.J. [ed.] Y. Dodge. North-Holland : s.n., 1987. pp. 405416.

[16] X-means: Extending K-means with Efficient Estimation of the Number of Clusters, D. Pelleg and A. Moore, in ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, 2000.

[17] 25. Workshop on Graph Theory, B. Stein and O. Niggemann, Ascona: LNCS Springer, 1999.

[18] Recent Advances in Graph Partitioning. A. Buluc, H. Meyerhenke, I. Safro, P. Sanders and C. Schulz, Springer International Publishing, 2016, pp. 117-158.