https://doi.org/10.3311/PPee.23363

Periodica Polytechnica Electrical Engineering and Computer Science, 68(4), pp. 319–326, 2024

Algorithmic Decomposition of Railway Objects for Distributed Interlocking System

Péter Arató[1], Tibor Gergely Markovits[1*], György Rácz[1]

[1] Department of Control Engineering and Information Technology, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Magyar tudósok krt. 2., H-1117 Budapest, Hungary
[*] Corresponding author, e-mail: markovits@iit.bme.hu

Received: 04 September 2023, Accepted: 16 August 2024, Published online: 13 September 2024

Abstract

Railway interlocking systems can be implemented as distributed systems, where each part of a station is handled by a separate logical unit. The logical units of such systems form a network and communicate by interchanging messages. Such distributed architectures are well known in large industrial control systems. There are several design practices and also algorithmic task partitioning methods that are applicable in distributed control systems. Some of such methods can also be adapted in designing of railway interlocking systems as well. In the case of such systems, the communication time between components must be kept low. Namely each separate controller in a given route must be able to exchange their internal state within a limited time in order to permit the train movement authorization. This limitation could cause high traffic load, if every logical unit would be interconnected with each other. Therefore, the main goal of the minimization is to reduce the number of connections between logical units. This can achieved by distributing and assigning the topological railway objects to certain logical units.

Keywords

decomposition, interlocking, partitioning, distributed logic

1 Introduction

A distributed electronic railway interlocking subsystem can have several different architectures, the main distinction between different products is whether their core logic is implemented in a distributed or centralized way. In general, the distributed core logic can be more scalable, than the centralized one, thus it enables to use them for larger stations, while centralized interlocking systems are suitable for smaller stations, track sections. For this paper, we considered both homogeneous and heterogenous distributed architectures, where the main interlocking logic is implemented using several uniform, general purpose, internally redundant and fault tolerant logical units referred to as LG units or racks. In the homogeneous case, we assume, that the program execution time is similar for all types of railway objects. In the heterogeneous case, we also consider the difference in the program execution time. The logical units are interconnected by point-to-point network links.

The distributed system is responsible for controlling all the railway objects of a station. The individual railway objects such as light signals and switches are handled on a topological unit basis. The topological units (TUs) are sets of railway objects arranged according to the topology of the station. This means that each separate TU, for example a track or turnout encapsulates all kinds of its own light signals (main or shunting) and possibly occupancy sensors. In the case of axle counters, the counter logics are integrated in the TUs and the train detection points transmit the data to each affected TU.

In this sense there are four different TU to be handled:

- Track section
- Turnout
- Interval connection
- Buffer stop.

These four TUs are to be handled by so called applications on the proposed general purpose LG unit. Further on, it will be estimated that each LG unit will be able to handle at most some of these applications. This limit is a mandatory requirement for the application distribution algorithm. Therefore, for any given station all the TUs must be assigned to LGs in a specific way, algorithmically. Such distribution problems are usually considered as a

Cite this article as: Arató, P., Markovits, T. G., Rácz, G. "Algorithmic Decomposition of Railway Objects for Distributed Interlocking System", Periodica Polytechnica Electrical Engineering and Computer Science, 68(4), pp. 319–326, 2024. https://doi.org/10.3311/PPee.23363

form of graph decomposition problem, similar to those, used in high level synthesis applications [1]. For this purpose, a topology description graph should be constructed according to the actual topology of the station.

To the best of our knowledge, no decomposition algorithms exist, that were specifically designed for the purpose of railway interlocking load distribution. The spectral clustering is a well-known decomposition method for image processing and also for task decomposition in grid computing and in system level synthesis. The spectral clustering calculates the spectrum of the graph in order to decide which nodes belong to the same segment. The spectrum of the graph is basically the result of a linear mapping. This associates every graph node with a set of real numbers. Usually, in graph segmentation only one dimensional spectrum is used which means that every graph node is associated with a single real number. The differences between such numbers directly correspond to the "closeness" of the nodes. The more two nodes are connected, the less is the difference between their spectral values.

Distribution of the TUs may be achieved by applying the spectral clustering on the actual topology description graph of the station.

2 Methods

The topology description graph (TDG) is an undirected graph with simple, unweighted edges and nodes. The nodes of the graph represent TUs, while the edges represent the connections between them. Only the neighbouring objects are connected by edges.

A simple example station is shown in Fig. 1 and its TDG in Fig. 2.

The purpose of the decomposition is to create such a so called segment graph by grouping the nodes for forming the segments. The nodes of this segment graph will correspond to the LG units, and the edges to the actual communication channels between LG units. The decomposition of the TDG can be performed by various methods [2–7]. However, most of these methods used in typical high level synthesis applications, mainly operate on directed graphs, so they may not be universally applied to an undirected graph. Furthermore, most of the existing techniques [3, 5, 6] rely on mainly iterative methods, and therefore they are not suitable for practically large graphs. On the other hand, the spectral clustering methods [2, 4, 7] such as our proposed solution can provide a closed algebraic form for the high level problem.

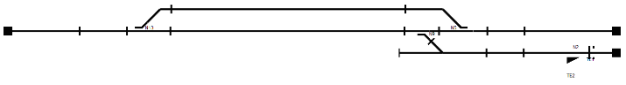

Fig. 1 A simple station topology

First, the set of metrics and rules must be defined by which the decomposition should be performed. The most important rule is that each segment can contain at most 4 TUs, since this is the limit of the considered architecture. Second, there are two metrics that must be optimized. The number of outgoing connections from each LG unit must be minimized. This means the maximal degree of the segment graph. Then, the number of LG units communicating on the longest possible route must also be minimized.

2.1 Normalized Spectral clustering

In the case of homogeneous distributed logic, when the other properties of the nodes are similar we can use the simple Normalized Spectral clustering.

If the data flow graph is only available in the form of an incidence matrix, it can be simply transformed into an adjacency matrix with the following transformation:

$$\boldsymbol{W} = \boldsymbol{B}(G)^T \boldsymbol{B}(G) \tag{1}$$

where $\boldsymbol{B}$ is the incidence matrix and $\boldsymbol{W}$ is the adjacency matrix of graph G. For any railway station, determining the incidence matrix is a systematic and straight forward process, as it can be easily generated from any graph description language.

In the following chapters it is assumed, that the vertices (V) of graph $G = (V, E)$ can be decomposed into two disjoint sets: A, and B, where it is true, that $A \cup B = V$ and $A \cap B = \varnothing$ by removing certain edges (E), that connected the two parts.

By definition, the Normalized Cut [8] is a cut minimalization process with a closed formula, that can consider the isolated nodes against the whole graph:

$$\mathrm{Ncut}(A,B) = \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(A,V)} + \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(B,V)}, \tag{2}$$

where

$$\mathrm{assoc}(A,V) = \sum_{u\in A, t\in V} w(u,t) \tag{3}$$

and

$$\mathrm{cut}(A,B) = \sum_{u\in A, v\in B} w(u,v). \tag{4}$$

The spectral clustering calculates the spectrum of the graph in order to decide which nodes belong to the same segment. The spectrum of the graph is basically the result

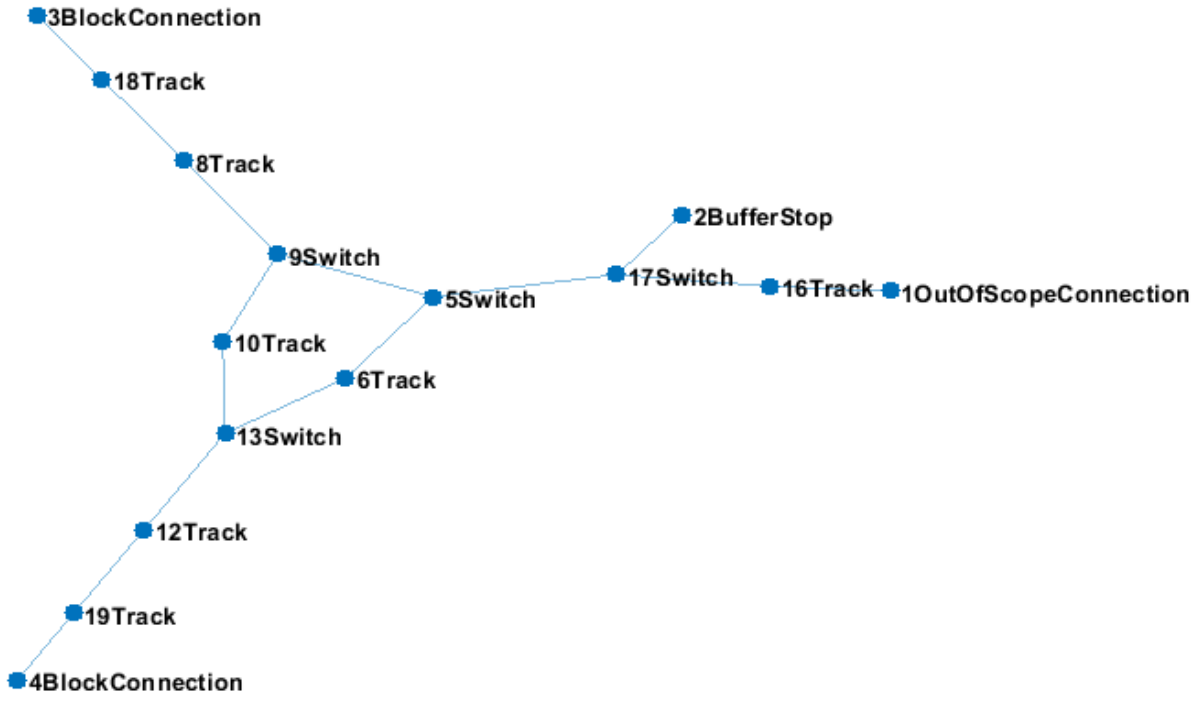

Fig. 2 A simple station topology

of a linear mapping. This associates every graph node with a set of real numbers. Usually, in graph segmentation only one dimensional spectrum is used which means that every graph node is associated with a single real number. The differences between such numbers directly correspond to the "closeness" of the nodes. The more two nodes are connected, the less is the difference between their spectral values. In order to calculate the spectrum, first we must construct the Laplacian matrix (L) of the graph, since from Eq. (2) it can derived to the form of:

$$(\boldsymbol{D}-\boldsymbol{W})\,\boldsymbol{y}=\lambda \boldsymbol{D}\boldsymbol{y}. \tag{5}$$

where $\boldsymbol{D}$ is a diagonal matrix with $\boldsymbol{d}$(i), where $\boldsymbol{d}$(i) is

$$\boldsymbol{d}(i)=\sum\nolimits_j w(i,j), \tag{6}$$

λ is the main objective of the minimization and $\boldsymbol{y}$ is the vector, that will determine whether a specific node belongs to group *A* or *B* as can be seen in [7].

Note, that the generalized eigenvalue system above is a Rayleigh quotient, and we can minimize it by solving the equation. By reorder the Eq. (5), the Laplacian matrix is the following:

$$\boldsymbol{L}=\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{D}^{-\frac{1}{2}} \tag{7}$$

Since L is positive-semidefinite, it always has non negative eigenvalues.

$$\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{z}=\lambda \boldsymbol{z} \tag{8}$$

The eigenvector corresponding to the smallest non zero eigenvalue is usually considered as the one-dimensional spectrum:

$$\boldsymbol{y}_1=\arg\cdot\min_{\boldsymbol{y}^T\boldsymbol{D}1=0}\left(\frac{\boldsymbol{y}^T(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y}}\right). \tag{9}$$

As an example, Table 1 shows the first three eigenvectors corresponding to the three smallest eigenvalues that can be derived of the graph in Fig. 2.

The one-dimensional spectrum (y1 column from Table 1) of the TDG related to the simple station shown in Fig. 3.

The spectrum can also be calculated in multiple dimensions. In this case, a vector is associated with each node, instead of a real number. The coordinates of these *N* dimensional vectors are the coordinates of the *N* eigenvectors corresponding to the first *N* smallest non zero eigenvalues. In the case of multidimensional vectors, the distances between points can still be calculated based on the Euclidean distances of the spectrum vectors.

The 2-dimensional spectrum of the TDG in Fig. 2 is shown in Fig. 4.

The 3-dimensional spectrum of the TDG in Fig. 2 is shown in Fig. 5.

Table 1 Solution of the eigenvalue problem of Fig. 2

TU identifier	y1	y2	y3
1OutOfScopeConnection	0.4057	0.2797	−0.3848
2BufferStop	0.2901	0.1879	0.0868
3BlockConnection	0.0385	−0.5739	−0.3313
4BlockConnection	−0.4967	0.2320	−0.3840
5Switch	0.0796	0.0292	0.2531
6Track	−0.0347	0.0416	0.3486
9Switch	0.0112	−0.1180	0.2508
8Track	0.0239	−0.3274	0.0418
10Track	−0.0713	−0.0380	0.3471
13Switch	−0.1444	0.0477	0.2907
12Track	−0.3080	0.1324	0.0485
17Switch	0.2516	0.1596	0.0486
16Track	0.3520	0.2376	−0.2155
18Track	0.0334	−0.4874	−0.1855
19Track	−0.4309	0.1971	−0.2151
Eigenvalues	0.1324	0.1508	0.44

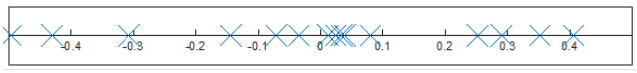

Fig. 3 One dimensional spectrum of the simple station

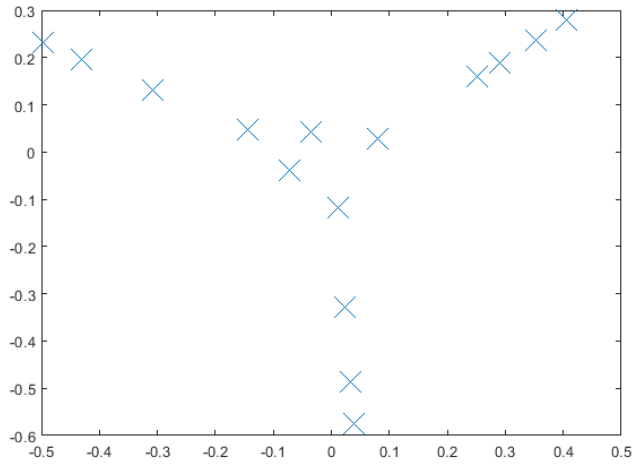

Fig. 4 Two dimensional spectrum of the simple station

After the spectrum is calculated for a specific TDG, the spectrum values are used as input to a multidimensional k-means algorithm, which sorts the values into clusters based on Euclidian distance [9].

Since k-means may be used in multiple dimensions, it was found that the 3 dimensional spectrum was usually enough to provide a suitable clustering, therefore the first 3 eigenvectors were chosen only. The main problem in k-means is that the resulting cluster sizes will not be

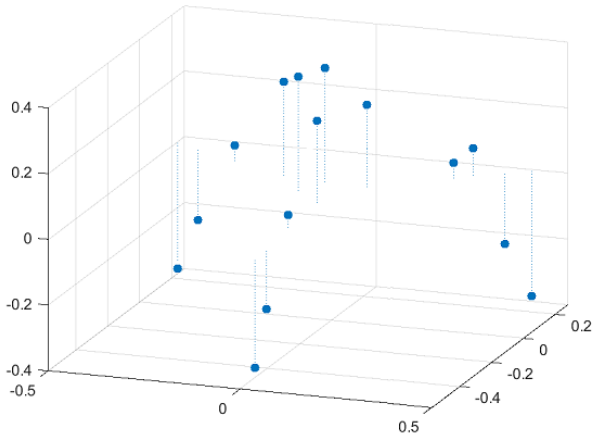

Fig. 5 Three dimensional spectrum of the simple station

uniform. There are some solutions for constraining the minimum cluster size for k-means [10], one is already implemented in Matlab [11]. This kind of constraint however does not fit our needs, since the TDG nodes will form the applications running on LGs. However, LGs have a limited computational capacity which is a maximum constrain on cluster sizes. Therefore, we've adapted the solution in [11] similarly to [12] to take into account the maximum cluster size instead, and create new clusters whenever an existing one would exceed the given maximum size.

The resulting algorithm can provide clusters close to but always less than the preferred maximum size.

2.2 Weighted Normalized Spectral clustering

In the case of the heterogenous version of distributed interlocking core logic, we should consider the different type of TUs take different execution time and different amount of resources to control. Therefore assigning the same number of TUs to every LG will not result a balanced distribution of load. This can be handled by adding weights to the nodes of the different types of TUs.

The NCut algorithm is originally only suitable for image segmentation in which the gradient between the pixels is represented by the weight of the edges. The nodes themselves in the original problem have no properties, therefore they are not represented in the normalized cut equation. This problem can be solved by using the Weighted Normalized Cut algorithm, that is a modified version of the original spectral clustering algorithm [13].

The main idea behind the Weighted Normalized Cut is extending the TDG with full subgraphs (cliques) on the desired nodes as seen in Fig. 6. This can be ensured by adding weight of the edges in the different cliques attached to the original nodes.

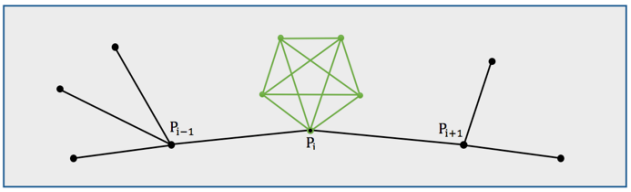

Fig. 6 Extension of the Pi node

The weight of the edges in the full subgraphs can be determined by using the trace of the extended diagonal matrix $\boldsymbol{D}_{ext}$, the weights of the nodes and therefore the number of edges in the full subgraphs as the following:

$$w_{jk,K_i} = \frac{2(\max_i v(i)-1)\cdot tr\left(\boldsymbol{D}_{ext}\right)}{\left|E_{K,i}\right|-1}. \tag{10}$$

After extending the original TDG, the generated graph can be called ETDG (extended topology description graph). The Normalized Spectral Clustering can be applied to the ETDG, thus the following steps are identical to the previous steps, that described in Section 2.1.

In the following example, the station is the same as it was before, the layout shown in Fig. 1. Additionally, the weights for the different types of TUs are:

- Track section: 4
- Turnout: 3
- Interval connection: 1
- Buffer stop: 1.

The ETDG of Fig. 1 is shown in Fig. 7.

The spectrum of the ETDG is shown in Fig. 8 which is similar, but not identical as the original one, without weights. The differences are definitely visible between Fig. 3 and Fig. 7, because TUs with weight of 1 are much closer to other TUs. Weights are also taken into account by the modified k-means clustering algorithm.

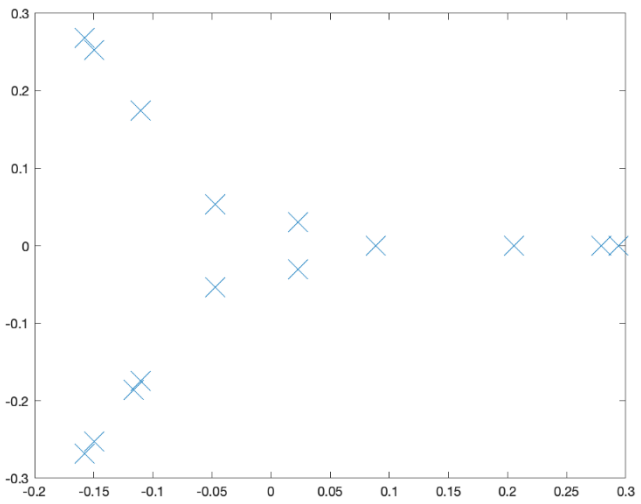

Fig. 7 The ETDG of Fig. 1

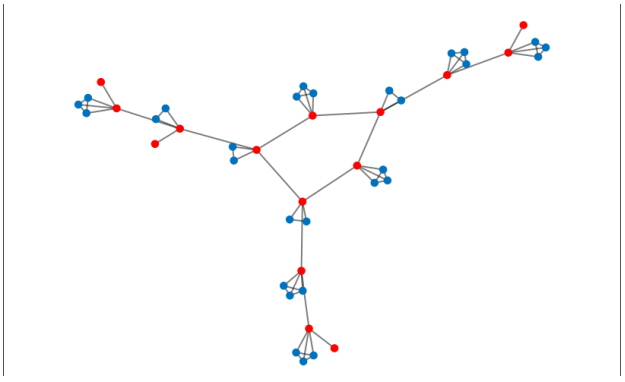

Fig. 8 The spectrum of the ETDG

3 Results

In order to test the method on an actual large example, Budapest Kelenföld railway station was chosen as it is one of the busiest and most complex railway stations in Hungary.

The topology of this station is shown in Fig. 9.

In the case of this large example The number of TUs is 244 and the number of required LG units is 61. The resulting decomposition is shown in Fig. 10.

The metrics for this decomposition show that the maximal number of communication channels between any two LG units is 8 which is sufficiently low. Also, the longest communication path contains 10 LG units, but can be further improved simply by looking at the resulting graph.

In order to further improve the decomposition for such large stations, a simple local refinement method was applied as shown below.

3.1 Local refinement

Many decomposition and most hardware-software partitioning methods apply some sort of simple local refinement algorithm [13] in order to improve the initial results of a more complex global algorithm. By analysing the results of the above decompositions, it is apparent that simply by swapping some TUs between LG units will provide better results.

Based on this idea a simple local refinement strategy can be used:

1. Take a list of all pairs of TUs that are assigned to different LG units.
2. Check each item (TU pair) on the list in sequence and do the following iteration:
   a. If neither TU in a given pair have neighbours in the other segment, do nothing and try another pair.
   b. Else, exchange the two TU between the segments.

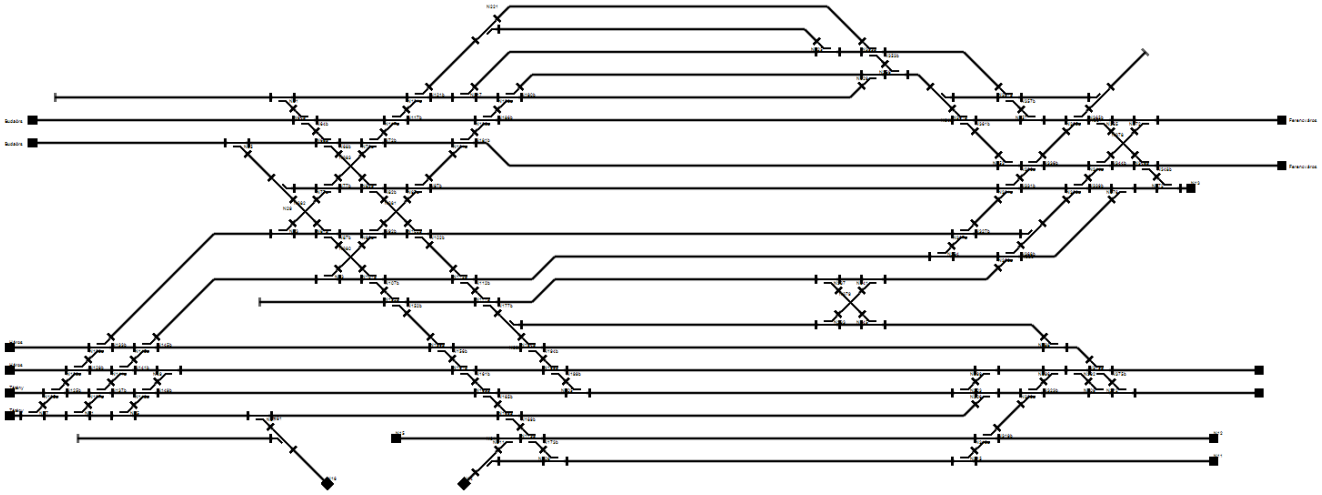

Fig. 9 Kelenföld Railway station

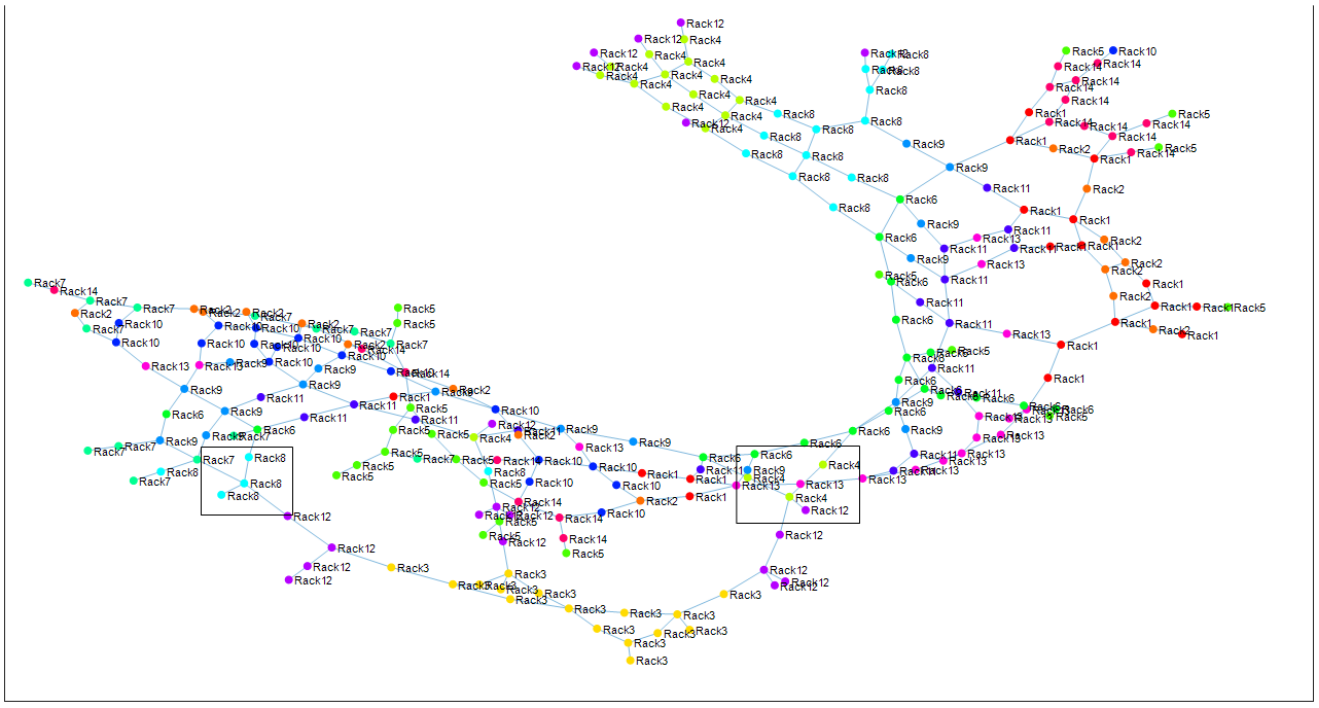

Fig. 10 TDG of Kelenföld Railway station

c. Recalculate metrics, and undo the exchange if neither metric improves or either of them gets worse because of it. Otherwise keep the TUs exchanged.

d. If the exchange was not withdrawn, recreate the whole list and start a new iteration.

3. Stop if all the exchanges were withdrawn in an iteration.

Obviously, this algorithm cannot guarantee the globally optimal result and can take much time on a randomly partitioned graph. However, after an aforementioned spectral decomposition has already been performed, only a few of the possible exchanges will improve the metrics. Because of this, the execution time of this algorithm will not be excessive.

3.2 Results of the local refinements

As an illustrative example of the local refinement, Fig. 11. shows the result of the weighted spectral clustering methods, where the TUs are allocated to 4 LGs (rack1 through rack4).

It can be seen on Fig. 12, that locking a route through this station requires communication between all four LG units. However a much better solution can be obtained just

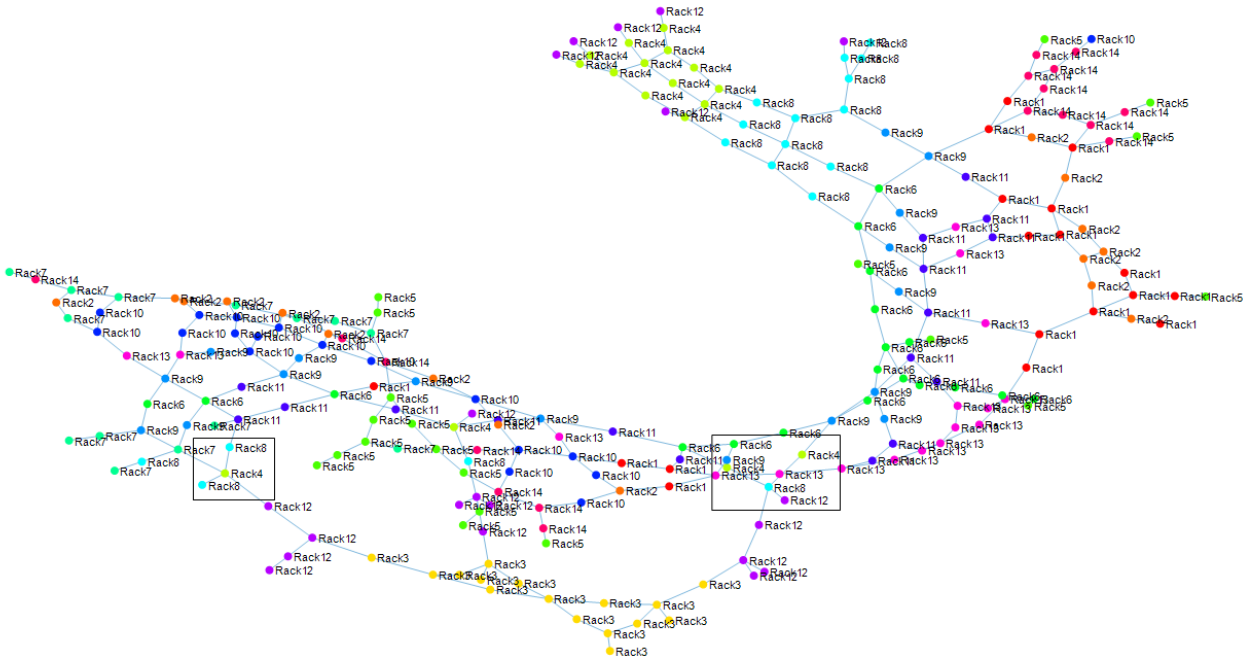

Fig. 11 Refinement of the TDG, where the most visible differences are marked with the black boxes

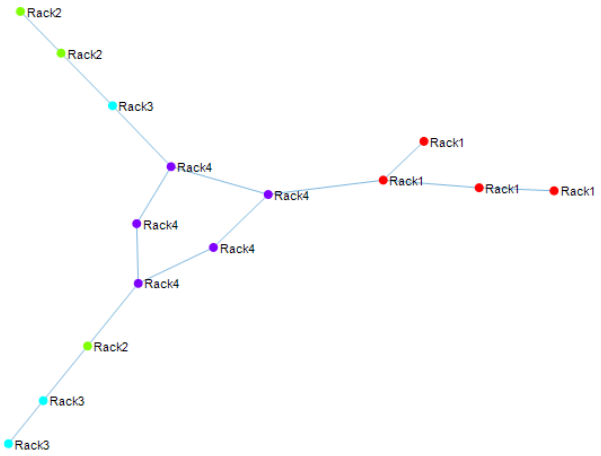

Fig. 12 Weighted normalized cut result for a simple station

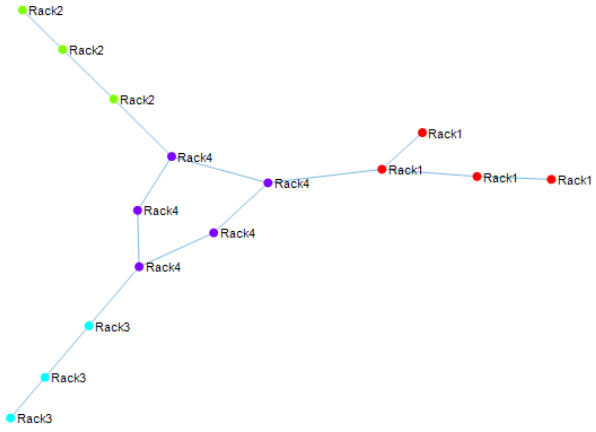

Fig. 13 Result of the local refinement applied on Fig. 12

by swapping the LG assignment of just two TUs. Such a solution is provided by our local refinement algorithms, and the end result can be seen in Fig. 13.

Similarly, the results of the large railway station of Kelenföld can be seen after this local refinement algorithm in Fig. 11.

Applying the local refinement to the previous example reduces the maximal number of communication channels from 8 to 7 and also reduces the longest communication path from 10 to 8 LG units. Thus, only 442 exchanges were attempted comparing with the possible 59292 random combinations.

The resulting decomposition for this station could not be improved further by trivial measures. The execution time of these algorithms implemented in MATLAB on a contemporary PC (Intel i7 2640m, 8GB RAM) required approximately 24 min from which the local refinement needed 22 min. This is because the spectral clustering algorithm simplifies most parts of the minimization in an exact mathematical formula, where only the eigensolver requires iteration.

4 Conclusion

We have proposed a new algorithm for task partitioning of distributed railway interlocking subsystems. The new

algorithm uses the multidimensional spectral clustering method completed with a new local refinement. This algorithm can provide close to optimal decompositions with respect to the defined metrics even for station with practical size. The execution time of the algorithm could still be optimized by a better implementation, although it provides results in an acceptable time-frame for typical station sizes.

The use of distributed logic in an interlocking system is very important, because of the railway RAM (reliability, availability, maintainability) requirements, described in EN 50126-1:2018 [14]. The distributed logic is proven more reliable then a centralized interlocking system, as a fallback of a single LG unit won't cause shutdown of the entire station.

Properly implemented distributed logic [15] will ensure a highly scalable interlocking system, because the same kind of LG units can be used in different quantities depending on the station size and topology. This algorithm can provide a cost effective interlocking logic distribution method for the locking dependencies.

References

[1] Arató, P., Drexler, D., Rácz, Gy. "Analyzing the Effect of Decomposition Algorithms on the heterogeneous Multiprocessing Architectures in System Level Synthesis", Scientific Bulletin of Politechnica University of Timisoara Transactions on Automatic Control and Computer Science, 60(74), pp. 39–46, 2015.

[2] Hendrickson, B., Leland, R. "The Chaco User's Guide Version 2.0", Sandia National Laboratories, Albuquerque, NM, USA, Technical Report SAND94-2692, 1994. https://doi.org/10.2172/10106339

[3] Hendrickson, B., Leland, R. "A Multilevel Algorithm for Partitioning Graphs", In: Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing, San Diego, CA, USA, 1995, 28. ISBN 978-0-89791-816-9 https://doi.org/10.1145/224170.224228

[4] Hendrickson, B, Leland, R. "An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations", SIAM Journal of Scientific Computing, 16(2), pp. 452–469, 1995. https://doi.org/10.1137/0916028

[5] Karypis, G., Kumar, V. "Metis - Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0", University of Minnesota, Minneapolis, MN, USA, 1995.

[6] Trifunovic, A., Knottenbelt, W. J. "Parkway 2.0: A Parallel Multilevel Hypergraph Partitioning Tool", In: Aykanat, C., Dayar, T., Körpeoğlu, İ. (eds.) Computer and Information Sciences - ISCIS 2004, Kemer-Antalya, Turkey, 2004, pp. 789–800. ISBN 978-3-540-23526-2 https://doi.org/10.1007/978-3-540-30182-0_79

[7] Rácz, Gy., Arató, P. "A Decomposition-Based System Level Synthesis Method for Heterogeneous Multiprocessor Architectures", In: Alioto, M., Li, H., Becker, J., Schlichtmann, U., Sridhar, R. (eds.) 2017 30th IEEE International System on Chip Conference (SOCC), Munich, Germany, 2017, pp. 381–386. ISBN 978-1-5386-4034-0 https://doi.org/10.1109/SOCC.2017.8226082

[8] Shi, J., Malik, J. "Normalized Cuts and Image Segmentation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), pp. 888–905, 2000. https://doi.org/10.1109/34.868688

[9] Lloyd, S. "Least squares quantization in PCM", In: IEEE Transactions on Information Theory, 28(2), pp. 129–137, 1982. https://doi.org/10.1109/TIT.1982.1056489

[10] Bradley, P. S., Bennett, K. P., Demiriz, A. "Constrained k-means clustering", Microsoft Research, Redmond, WA, USA, Technical Report MSR-TR-2000-65, 2000.

[11] Polk, S. "Constrained K-Means (1.0.0)", [computer program] Available at: https://www.mathworks.com/matlabcentral/fileexchange/117355-constrained-k-means [Accessed: 19 September 2022]

[12] Ganganath, N., Cheng, C., Tse, C. K. "Data Clustering with Cluster Size Constraints Using a Modified K-Means Algorithm", In: 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Shanghai, China, 2014, pp. 158–161. ISBN 78-1-4799-6236-5 https://doi.org/10.1109/CyberC.2014.36

[13] Arató, P., Markovits, T. G. "Utilizing the spectral properties of the data flow graph in system level synthesis", In: Szirmay-Kalos, L. (ed.) Proceedings of the Workshop on the Advances of Information Technology 2020: WAIT 2020, Budapest, Hungary, pp. 154–159. ISBN 9789634218029

[14] Comité Européen de Normalisation Electrotechnique "EN 50126-1:2018, Railway Applications – The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)", CEN-CENELEC, Brussels, Belgium, 2018.

[15] Markovits, T. G., Arató, P., Rácz, Gy. "Implementation of an SoC architecture with built-in safety features", In: 2021 IEEE 34th International System-on-Chip Conference (SOCC), Las Vegas, NV, USA, 2021, pp. 95–100. ISBN 978-1-6654-2931-3 https://doi.org/10.1109/SOCC52499.2021.9739573