УДК 621.396:004.75:004.78 **doi: 10.26906/SUNZ.2025.2.255-262**

Illia Syvolovskyi[1], Volodymyr Lysechko[2]

[1] Ukrainian State University of Railway Transport, Kharkiv, Ukraine
[2] Ivan Kozhedub Kharkiv National University of Air Forces, Kharkiv, Ukraine

A METHOD OF HIERARCHICAL CLUSTERING OF NODES IN DISTRIBUTED TELECOMMUNICATION SYSTEMS USING GRAPH ALGORITHMS

Abstract. This article describes a modified method for hierarchical clustering of computing nodes in distributed telecommunication systems, considering node performance, network topology, delays, and communication channel bandwidth. The proposed method is based on a modified Louvain algorithm that performs multi-step graph clustering with dynamic parameter adjustment. This makes it possible to control the size of clusters and their internal density in accordance with the specified targets, minimizing the fragmentation of the network structure. Based on a comparative analysis of modern clustering methods and experimental modeling, it has been found that the proposed method reduces cluster fragmentation by 36% compared to the Leiden method. In addition, it reduces inter-cluster delays by 4,5% compared to the Louvain method and by 11,8% compared to Leiden, which indicates a more efficient organization of inter-cluster interaction. The improved method ensures an even distribution of computing nodes among clusters, which helps to optimize the overall performance of a distributed telecommunications system.

Keywords: distributed telecommunication systems, hierarchy, cluster, node, modeling, intelligent systems, graphs, algorithms, optimzization, Louvain, Leiden, throughput, topology, delay minimization, dynamic self-organization,

Introduction

Statement of a scientific problem. With the increasing complexity of distributed computing systems and telecommunication networks, there is a growing need to efficiently organize computing nodes and minimize data transmission delays. The main challenges include the heterogeneity of node performance, variability of network connection parameters (latency, bandwidth), and dynamic changes in the network topology that affect the stability and performance of distributed computing processes.

Traditional clustering approaches do not fully take into account the specifics of the network topology and connectivity parameters, which can lead to irrational workload distribution and increased inter-cluster delays. This is especially significant in telecommunications systems, where the efficient organization of computing nodes affects the performance of the entire network.

Thus, there is a need to develop a clustering method that takes into account not only the modularity of the graph but also the key characteristics of connectivity between nodes, which will reduce latency, improve load balancing, and ensure adaptability in changing network conditions.

Research analysis. An analysis of modern clustering methods [1-18] confirms the effectiveness of graph algorithms for organizing distributed telecommunication systems. Studies [3, 5, 6, 7, 10, 11] confirm the effectiveness of distributed computing, but they are mostly focused on static models. In [14, 15], methods for minimizing delays are studied, but without taking into account dynamic changes in the network topology.

Studies of load balancing [1, 12] are related to mobile environments, but do not cover the issue of flexible node clustering.

The Louvain clustering method [2, 4] proves effective in detecting clusters, but has limitations in terms of resolution and cluster separation. In [14], the Leiden algorithm is proposed, which eliminates this drawback, but is computationally more complex.

Service placement methods [6, 7, 10, 16] are aimed at reducing data processing delays, but do not take into account the variability of nodes and the physical limitations of network topologies. Some cloud-based methods [9] have potential application in distributed systems (at higher processing levels close to the cloud), but are not adapted to clustering, which makes it impossible to effectively apply them on a large number of nodes.

Thus, the existing methods do not provide a comprehensive solution to the problem of adaptive clustering considering the network parameters. This necessitates the development of a new method that combines graph optimization and adaptive grouping of nodes according to specified targets.

The purpose of the work is to develop a method for hierarchical clustering of computing nodes in distributed telecommunication systems based on a modified Louvain algorithm with adaptive parameter adjustment. This ensures the formation of clusters of optimal size and connectivity, contributing to a balanced allocation of resources and minimizing inter-cluster delays.

Presentation of the main material and substantiation of the obtained research results

Considering the principles of multi-level telecommunication systems and the analysis of modern clustering methods [2, 4, 6, 12, 14], a hierarchical clustering method of nodes aimed at optimizing the network structure has been developed. The features of the proposed clustering method include:

1. Task-Oriented Clustering. Nodes are grouped according to the tasks they can perform [7, 10]. Each node can efficiently handle a certain task or class of tasks due to its hardware characteristics. The clustering method provides a way to form groups of nodes that can jointly perform one or more tasks, taking into account their performance.

2. Hierarchical (multi-level) structure of clusters (Hierarchical Cluster Organization). The clustering method involves multi-level clustering [3, 5, 12], due to the iterative separation of the graph structure with adaptive parameter settings.

3. Optimization of intra-cluster and inter-cluster interaction. It is focused on minimizing delays inside and outside the clusters, which is important when processing data in real-time systems [14, 15].

4. Use of weighted link parameters (Latency & Bandwidth-Aware Clustering). Since the geographical location of the nodes is not defined, the optimization of interaction is based on the latency and bandwidth of communication channels [9, 16]. The results of comparing the proposed clustering method with the reference method substantiated by scientists in [8] are presented in Table 1. The proposed hierarchical clustering method provides increased adaptability, efficient resource allocation, and better scalability compared to the reference method.

The implementation of the hierarchical clustering method begins with the definition of the cluster formation algorithm [2,4,14]. In order to substantiate the choice of the optimal clustering algorithm, which is the most relevant, a comparative analysis of modern approaches was conducted (Table 2) [5, 12, 13].

Table 1 – **Comparison of clustering methods**

Criterion	The proposed method	Reference method [8]	Advantage
Clustering type	Multi-level hierarchical clustering of nodes by task type	Two-stage clustering of communities with service placement	Easier and faster network organization and resource allocation
Complexity	Uses Louvain algorithm to find communities	Uses Girvan-Newman algorithm to find communities	Significantly lower computational complexity and higher execution speed
Weight parameter	Combined (delay + bandwidth)	Hop count	Optimally takes into account the physical characteristics of the network
Scalability	Adaptive hierarchical structure	Significant dependence on community sizes	Efficient work in large systems
Delay optimization	Local clustering + relative geographic optimization	Transient closure of services	Lower transmission delays between nodes
Prioritization of resources	Minimize intra and inter-cluster latency while sustaining target cluster sizes.	Uniform distribution of services between communities to satisfy deadlines	Faster resource management and task reassignment on failures

Table 2 – **Comparative characteristics of modern clustering algorithms**

Algorithm	Principle	Weight parameters	Adaptation	Advantages	Disadvantages
K-Medoids	Partitioning by similarity	Yes	Low	Weight-aware, emission-resistant	High complexity, not for large graphs
Multi-Level Partitioning	Recursive graph partitioning	Yes	High	Load balancing	Does not take into account the structure of clusters
Louvain	Optimization of modularity	Yes	High	Fast, scalable	Generates large clusters
Leiden	Improved Louvain algorithm	Yes	High	Fast, detailed partitioning, avoids resolution limit	Slower than Louvain, high computational complexity

As shown in Table 2, the most suitable for the scientific tasks of this study are the Louvain algorithms and its modification, the Leiden algorithm [2,4,14].

1. The Louvain algorithm was chosen because of its high speed and scalability, which is especially relevant for distributed computing systems. It optimizes the modularity of the graph, which allows for efficient community detection and cluster formation. Louvain's algorithm consists of the following steps (Fig. 1):

– initialization – each node is considered as a separate community;

– moving nodes – nodes are moved between communities to improve the quality function (e.g., modularity);

– community aggregation – nodes within one community are merged into a super-node that forms a new graph;

– repetition – the process is repeated recursively until stability is achieved.

The steps of the Louvain and Leyden algorithms are shown in Fig. 1 and Fig. 2. In the initial state (black and white image), all nodes are not assigned to any cluster. The color in the following stages indicates the membership of nodes in the corresponding clusters formed during clustering.

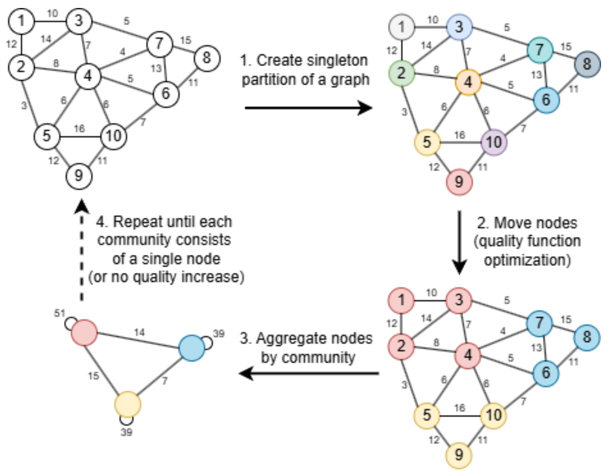
1. Create singleton partition of a graph
2. Move nodes (quality function optimization)
3. Aggregate nodes by community
4. Repeat until each community consists of a single node (or no quality increase)

Fig. 1. Stages of the Louvain algorithm

It has been studied that the main disadvantage of the Louvain algorithm is the tendency to form excessively large clusters, which does not meet the objectives of this study, since a more detailed breakdown of nodes is needed with a more accurate consideration of their internal structure [4]. The reason for this is that at each iteration, new communities are formed solely on the basis of the increase in modularity, which does not always guarantee the optimality of their partitioning.

2. In this research, the Leiden algorithm was chosen to compare the performance of the proposed hierarchical clustering method. The main advantage of the Leiden algorithm is the elimination of the main disadvantages of the Louvain algorithm, namely, resolution limit and cluster post-processing [14].

Unlike Louvain, Leiden clustering performs not only global optimization updates but also intermediate refinement at the level of individual nodes, which provides more detailed grouping.

The process of the Leiden algorithm consists of the following steps:

– initialization – each node is considered as a separate community;

– moving nodes – nodes are moved between communities to improve the quality function;

– cluster refinement – an additional stage of verification and adjustment of the division is applied;

– aggregation – nodes within the community are combined into a single node;

– repetition – the procedure continues until a stable partition is achieved.

The Leyden algorithm eliminates the problem of insufficient cluster detail present in the Louvain algorithm, as it allows for a more accurate distribution of nodes between clusters and prevents weakly connected structures from remaining in the same cluster (Fig. 2).

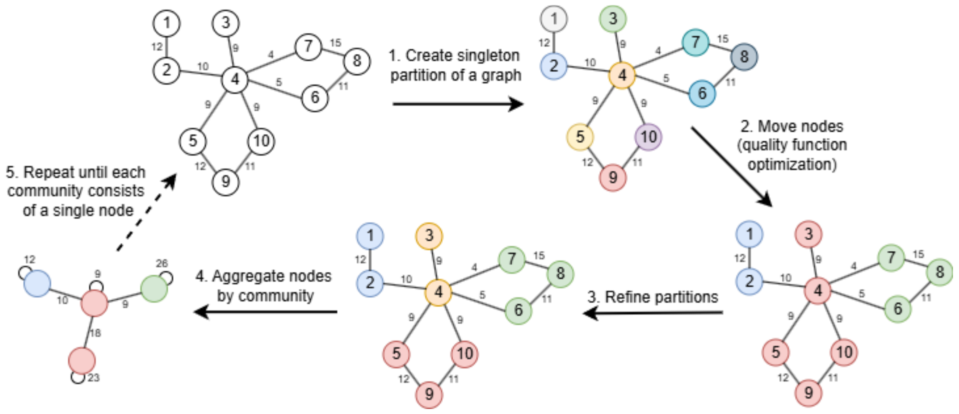
1. Create singleton partition of a graph
2. Move nodes (quality function optimization)
3. Refine partitions
4. Aggregate nodes by community
5. Repeat until each community consists of a single node

Fig. 2. Stages of the Leiden algorithm

This is achieved by introducing the resolution parameter γ into the modularity function, which allows to dynamically adjust the level of detail of the cluster structure in distributed telecommunication networks. This method provides an optimal balance between the accuracy of cluster detection and the stability of their structure, preventing excessive fragmentation or consolidation, which is critical for ensuring stable interaction between network nodes.

The proposed improvement of the Louvain algorithm allows us to develop a method of hierarchical clustering of nodes by improving the quality of clustering and optimal resource allocation.

The main advantages of the proposed method are: increased accuracy of cluster detection due to an adaptive approach to clustering; flexible configuration of clustering parameters, including target cluster size limits and cluster granularity (resolution) step; reduce inter-cluster latency by taking into account the delay and bandwidth between nodes.

The method includes the following steps.

Stage 1. Representation of a telecommunication system as a weighted graph $G = (V, E)$, where: V – a set of nodes corresponding to the computing elements of the system; E – a set of edges, links between nodes, characterized by certain weight parameters [6, 12]. Each node v_i has a set of properties, namely: label T_{ij}, indicating the type of task it can perform; computing performance P_{ij} of the node. Each edge e_{ij} between nodes v_i and v_j has the following parameters: w_{ij} – the bandwidth of the communication channel between i and j; d_{ij} – data transmission delay.

Since the Louvain and Leiden algorithms optimize the modularity of the graph, it is necessary to correctly determine the weight of the edges between nodes, which will simultaneously represent the delay and throughput. Different approaches to determining this weight are presented in [2,4,14].

1. Inverse throughput weighted by delay:

$$w(e) = \frac{d_{ij}}{w_{ij}}, \qquad (1)$$

It is used in cases where the main task is to minimize the delay.

2. Harmonic average of bandwidth and inverse delay:

$$w(e) = 2 \cdot w_{ij} \cdot \frac{1}{d_{ij}} \Big/ \left(w_{ij} + \frac{1}{d_{ij}}\right), \qquad (2)$$

Balances fast and slow links for heterogeneous networks but may overestimate links with very small delays d_{ij}.

3. Normalized communication quality evaluation:

$$w(e) = \alpha \cdot d_{ij} + \beta \cdot w_{ij}, \qquad (3)$$

Adjusts weights via α and β coefficients to adapt to changing network topologies and conditions. However, they must balance latency and throughput to avoid overestimating high-bandwidth links and reducing the effect of abnormal delays.

Approach 2 was selected, using a weighting function as a linear combination of delay and capacity. Based on these parameters, an initial graph is built, which will be used for clustering.

Stage 2. Splitting a graph into layers

At this stage, the nodes of the distributed system are classified by the types of tasks [6, 12] performed on the data during stream processing. This creates a multi-tiered structure, where each tier corresponds to a specific type of computing process. This approach simplifies further clustering, ensures localization of computing, and minimizes inter-cluster delays.

To classify nodes, we introduce a set of levels S of the computing process, where each level $s_k \in S$ is defined by a set of tasks $T(v_i)$ that can be performed by nodes of this level. It is determined by the criterion of task compatibility:

$$s_k = \{v_i | T(v_i) \cap L(s_k) \neq \emptyset \}, \qquad (4)$$

where $L(s_k)$ is the set of tasks assigned to level s_k.

This equation means that a node v_i is assigned to level s_k if its task matches at least one of the tasks assigned to that level.

Stage 3. Recursive clustering of each layer

Clustering at each level is performed using a modified Louvain algorithm that focuses on achieving target cluster sizes by recursively detecting them. The modularity formula used by the standard Louvain algorithm does not contain parameters that would allow adjusting the connectivity of the original clusters. To solve this problem, the algorithm uses a modified modularity formula, the Reichardt Bornholdt Potts Model, which defines the resolution parameter γ [2, 4, 14, 19]. By adjusting this parameter as the algorithm runs, a more detailed clustering can be achieved, since different values of the resolution parameter are applied to different parts of the graph. The modularity Q, considering γ, is calculated by the formula:

$$Q(v) = \frac{1}{2m} \sum_{i,j} \left[A_{i,j} - \gamma \frac{k_i k_j}{2m} \right] \delta(c_i, c_j), \qquad (5)$$

where γ – a parameter that controls the level of detail of the clusters; $\gamma > 1$ – promotes the creation of smaller clusters; $\gamma < 1$ – promotes the formation of larger clusters; $A_{i,j}$ – adjacency matrices that determine the existence of a connection between nodes; $k_i k_j$ – the degrees of nodes i and j (the total number of links in the graph); m – the total weight of all links in the graph; c_i, c_j – corresponding clusters for nodes i , j; $\delta(c_i, c_j)$ – an indicator of whether a node belongs to a cluster or not.

Recursive clustering based on the modified Louvain algorithm includes the following steps.

3.1 Initialize input data, parameters, and variables.

Input: graph G, partitioning quality function $Mod(\gamma)$, base value of the resolution parameter γ, resolution step γ_{step}, number of cluster search iterations N_{probe}, and cluster size limits: lower bound C_{low} and upper bound C_{up}. Internal variables: current resolution value Υ_{curr} and partitioning P_{curr}, cluster search iteration N_{curr}.

3.2 Execution of the standard Louvain algorithm.

The Louvain algorithm is applied to the current graph G, which uses the Reichardt Bornholdt Potts Model to calculate the quality and the parameter Υ_{curr} as a resolution parameter. This forms the initial partitioning of P_{curr}.

3.3 Checking the minimum cluster size.

If at least one resulting cluster is smaller than C_{low}, the Υ_{curr} parameter is reduced by the formula:

$$\Upsilon_{curr} = \max(\Upsilon - \frac{\Upsilon_{step}}{2^{N_{curr}+1}}, 0{,}5), \qquad (6)$$

This operation prevents excessive fragmentation of clusters and allows you to select the optimal resolution value for the current graph. The resolution value of 0.5 is the minimum threshold for the algorithm to work without showing abnormal behavior.

3.4 Checking the success of the cluster search.

If the cluster search was successful, the value of Υ_{curr} is increased by Y_{step}, and if not, the algorithm returns the current partition P_{curr} as a single community.

3.5 Recursive processing of large clusters.

For each obtained cluster, the size is checked: if the cluster size C_{sub} is smaller than C_{up}, it remains unchanged; if the size exceeds C_{up}, an induced subgraph G_{comm} is created containing only the nodes of this cluster. Next, LouvainRecursive is again executed on this subgraph, which allows for adaptive refinement of large clusters.

3.6 Aggregation of all obtained clusters into a single partition.

All the resulting partitions of the current level P_{comm} are combined into a single subpartition P, which returns to the level above until it reaches the final exit from the function.

3.7 Formation of the final cluster structure.

The resulting clusters of each layer are aggregated into the final cluster structure. Thus, the resulting partitioning considers the topological features of the network and its parameters, dynamically adapting the clustering to the specified size parameters.

The flowchart of the algorithm that implements the stages of the described method is shown in Fig. 3.

To evaluate the efficiency of the proposed hierarchical clustering method, we developed a software implementation of the Louvain, Leiden, and modified Louvain algorithms using the Python programming language and the NetworkX library. To increase the reliability of the experiment, a graph of 2000 nodes was generated, the structure of which is as close as possible to a real distributed telecommunications network [17]. To visualize the resulting graph and communities, the Fruchterman-Reinhold power algorithm was used to emphasize the visual grouping of nodes if a strong connection is identified between them [18].

The conducted modeling resulted in a graph representation of a certain distributed telecommunication system, which was used to divide the nodes into clusters according to different approaches (Fig. 4 A-C).

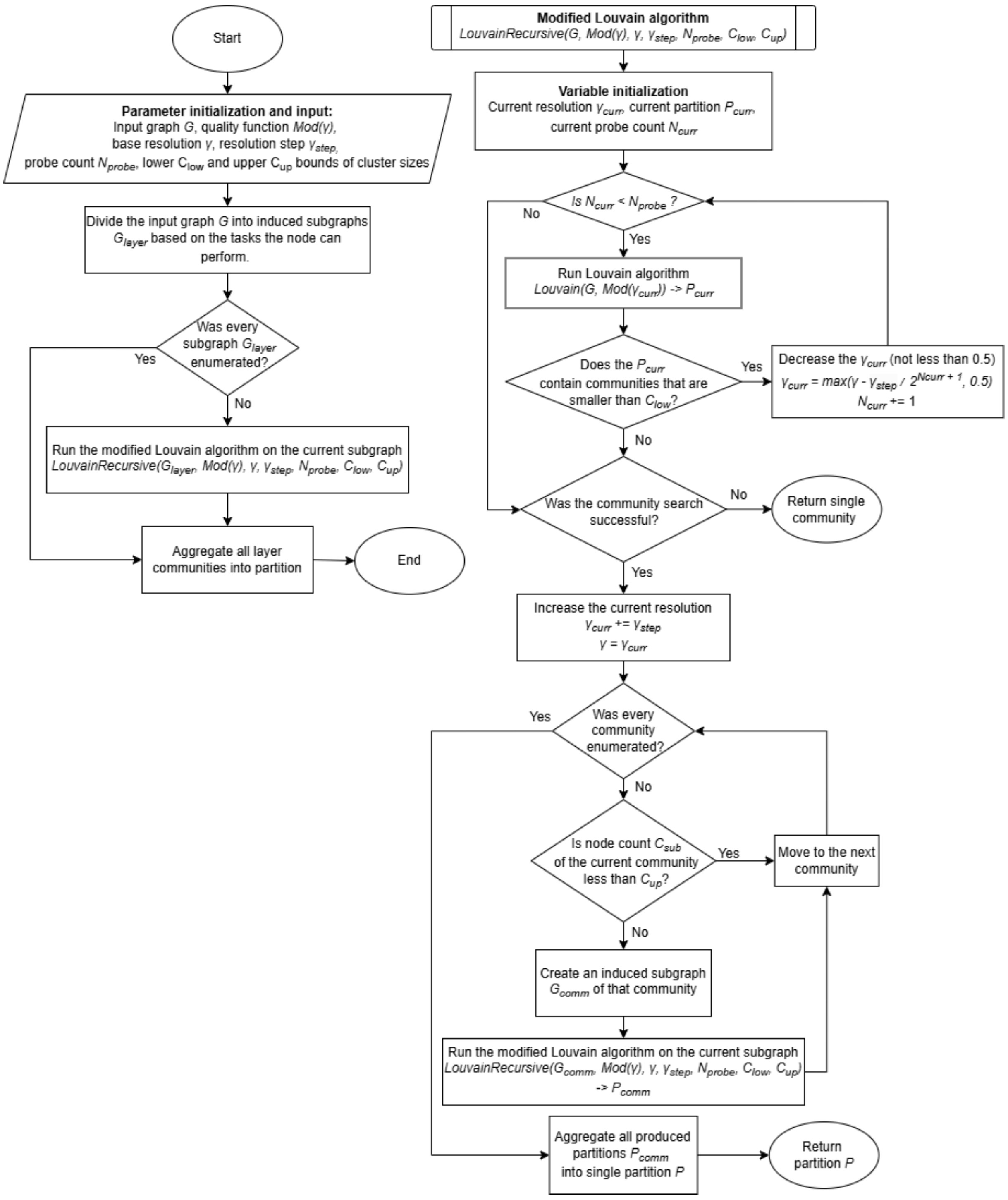
Start
Parameter initialization and input: Input graph G, quality function Mod(γ), base resolution γ, resolution step γstep, probe count Nprobe, lower Clow and upper Cup bounds of cluster sizes
Divide the input graph G into induced subgraphs Glayer based on the tasks the node can perform.
Was every subgraph Glayer enumerated?
Yes
No
Run the modified Louvain algorithm on the current subgraph LouvainRecursive(Glayer, Mod(γ), γ, γstep, Nprobe, Clow, Cup)
Aggregate all layer communities into partition
End
Modified Louvain algorithm LouvainRecursive(G, Mod(γ), γ, γstep, Nprobe, Clow, Cup)
Variable initialization Current resolution γcurr, current partition Pcurr, current probe count Ncurr
Is Ncurr < Nprobe ?
Run Louvain algorithm Louvain(G, Mod(γcurr)) -> Pcurr
Does the Pcurr contain communities that are smaller than Clow?
Decrease the γcurr (not less than 0.5) γcurr = max(γ - γstep / 2^(Ncurr + 1), 0.5) Ncurr += 1
Was the community search successful?
Return single community
Increase the current resolution γcurr += γstep γ = γcurr
Was every community enumerated?
Is node count Csub of the current community less than Cup?
Move to the next community
Create an induced subgraph Gcomm of that community
Run the modified Louvain algorithm on the current subgraph LouvainRecursive(Gcomm, Mod(γ), γ, γstep, Nprobe, Clow, Cup) -> Pcomm
Aggregate all produced partitions Pcomm into single partition P
Return partition P

Fig. 3. Flowchart of the modified hierarchical clustering method

In these figures, graph vertices (colored dots) correspond to computing or communication elements of the telecommunications system (servers, routers, network devices), and edges (lines of the corresponding color) correspond to data transmission channels, taking into account delays and bandwidth. The colors of the nodes reflect the cluster structure obtained as a result of clustering: nodes of the same color belong to the same cluster. Edges that have the color of the corresponding cluster identify intra-cluster connections, while black lines indicate inter-cluster connections between nodes from different clusters. Fig. 4.A shows a graph of a distributed telecommunication system with defined clusters, which was obtained as a result of clustering using the Louvain algorithm.

In this graph, the nodes correspond to computing or communication elements of the system (servers, routers, network devices), and the links model the data transmission channels between them, taking into account bandwidth and delays. The graph shows a cluster structure, where nodes with high interaction are grouped into separate communities (colored segments). In the center of the graph are clusters with dense connections, and on the periphery are less connected nodes that form long links with the central part. The main feature of Louvain clustering is the tendency to form large clusters, even if there is no strong enough connection between them. This is a consequence of the algorithm's greedy approach. The introduction of a resolution parameter provides a way to partially adjust the requirements for cluster connectivity, but does not completely eliminate the problem: at low values of the parameter, clusters remain too large, and at high values, they are unevenly split.

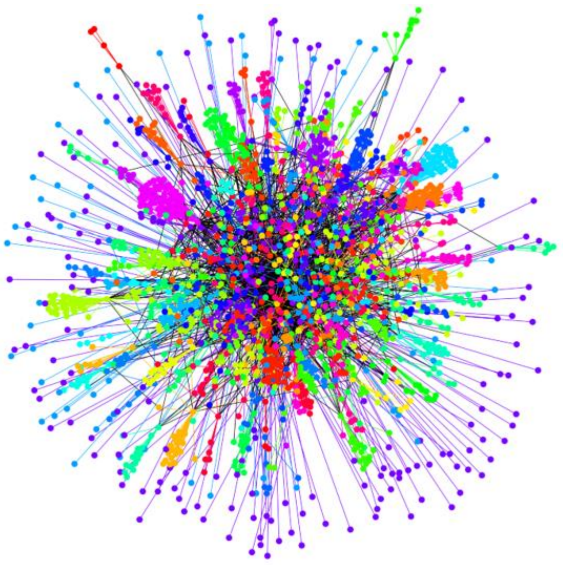

Fig. 4.A. Clustering of nodes using the Louvain algorithm

Fig. 4.B shows a graph of a distributed telecommunication system clustered by the Leiden algorithm.

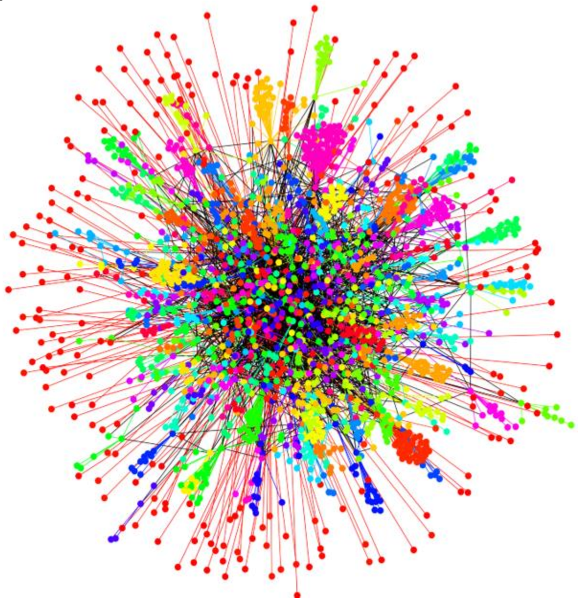

Fig. 4.B. Clustering of nodes using the Leiden algorithm

Fig. 4.B shows that the Leiden algorithm forms more evenly distributed clusters than Louvain, which reduces the centralization of the system structure. Nodes connected by long links to the central part of the graph remain on the periphery.

One of the key features of the algorithm is the way it works on graphs with a distinct cluster structure. Even at low values of the resolution parameter, Leiden finds a near-optimal partitioning, and its further increase causes only a slight growth of fragmentation.

However, the problem typical for the Louvain algorithm remains: clusters with a high density of connections may be separated unevenly or with a delay in the process of increasing the resolution parameter. This can lead to a situation where weakly connected areas of the graph are already divided into small clusters, while dense communities remain unchanged.

To solve these problems, a modified clustering method is proposed that adaptively changes the resolution parameter, preventing uneven distribution of clusters.

Fig. 4.C shows the visualization of the clustering of the graph of a distributed telecommunication system performed using the proposed modified method, which minimizes both intra-cluster and inter-cluster delays.

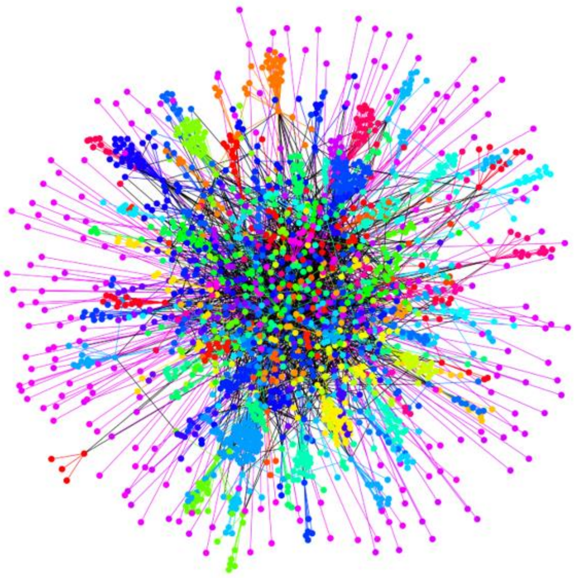

Fig. 4.C. Clustering of nodes using the modified method

Unlike the Louvain and Leiden algorithms, the proposed method forms balanced clusters, which prevents both excessive community consolidation and excessive fragmentation. This approach to clustering considers the actual characteristics of the telecommunications system and its requirements, including channel bandwidth, node performance, and target cluster sizes. This ensures stable interaction between clusters, reduces delays in inter-cluster data exchanges, and reduces decision-making time within the cluster.

To evaluate the effectiveness of the clustering methods, we conducted an experimental simulation, the results of which are shown in Table 3. The simulation was performed at a resolution value of $\Upsilon = 1$.

Based on the results of the experiment, the following conclusions can be drawn:

1. Louvain generates 50 clusters, which leads to large communities with the highest modularity (0.851892). However, the inter-cluster delays reach 111.928, which is 16.9% higher than Leiden and 4.6% higher than the modified method.

Table 3 – **Comparative analysis of clustering methods**

Parameter	Louvain	Leiden	Modified method
Number of clusters	50	215	137
Modularity	0,851892	0,811312	0,825021
Maximum cluster size	184	66	64
Minimum cluster size	4	2	2
Average cluster size	40,000	9,302	14,598
Standard deviation of cluster size	28,706	9,551	14,134
Average delay	165,607	165,607	165,607
Average intra-cluster latency	52,911	45,306	52,863
Average inter-cluster latency	111,928	95,644	106,930

2. Leiden generates 215 clusters (4.3 times more than Louvain), which reduces inter-cluster latency by 14.5% compared to Louvain. However, this is accompanied by cluster fragmentation, as their maximum size (66) is 64.1% smaller than that of Louvain (184), and the standard deviation of cluster size is 66.7% smaller.

3. The modified method generates 137 clusters, which is 174% more than Louvain, but 36% less than Leiden. This reduces inter-cluster latency by 4.5% compared to Louvain, avoiding the excessive granularity of Leiden. In addition, intra-cluster latency (52,863) remains almost at the level of Louvain (+0.1%), which guarantees system stability.

Thus, the proposed modified method combines a balanced distribution of nodes and minimal delays, which makes it optimal for distributed systems.

Conclusions and prospects for further research

The study presents a new method of hierarchical clustering of computing nodes in distributed telecommunication systems based on a modified Louvain algorithm. The method is aimed at optimizing modularity, intra- and inter-cluster delays, and cluster sizes, ensuring efficient resource allocation in the network.

1. It is experimentally proved that the proposed hierarchical clustering method and its implementation algorithm are versatile and provide opportunities for building various topologies, including those based on geo-distribution.

2. By increasing the resolution, communities with a larger number of clusters (of smaller size) are created. However, this process can occur unevenly: when using a large value of the resolution parameter (10-20 and above), it has been experimentally proven that clusters with size 1 are formed, which cannot be practically used, while clusters with high connectivity remain undivided. Their separation occurs at a much higher resolution parameter (at the level of 50-100), but at the same time, most of the graph will contain clusters of size 1-2.

3. The combination of the Louvain algorithm with a hierarchical structure allows clustering to be applied not only to the full network graph of the system, but also to its individual segments. This makes it possible to optimally rebuild some of the network segments without affecting those clusters that have already reached a sufficient level of optimization.

The scientific novelty of the study is the development of a new method of hierarchical clustering of computing nodes in distributed telecommunication systems, which, unlike traditional approaches, takes into account the performance of nodes, network topology, delays and bandwidth of communication channels, which is achieved by adaptively adjusting clustering parameters and multilevel analysis of network connections.

Further research is focused on improving the method by expanding its adaptive capabilities to work with heterogeneous telecommunication systems that include nodes with different architectures and computing capabilities.

References

1. Abdelmoneem Randa M., Benslimane Abderrahim, Shaaban Eman (2020) Mobility-aware task scheduling in cloud-Fog IoT-based healthcare architectures. Computer Networks. Volume 179, 9 October 2020, P.P 107348 - 107367 https://doi.org/10.1016/j.comnet.2020.107348
2. Blondel, V.D. et al. (2008) Fast unfolding of communities in large networks. J. Stat. Mech 10008, 1-12(2008). https://doi.org/10.1088/1742-5468/2008/10/P10008
3. Bonomi F. , Milito R., Natarajan P., and Zhu J. (2014). Fog Computing: A Platform for Internet of Things and Analytics. Springer International Publishing, Cham, 169-186. DOI:10.1007/978-3-319-05029-4_7
4. Dugué N., Perez A. (2015) Directed Louvain: maximizing modularity in directed networks. [Research Report] Université d'Orléans. 2015. DOI:10.13140/RG.2.1.4497.0328
5. Ferreira Aluizio, Neto Rocha (2021) Edge-distributed Stream Processing for Video Analytics in Smart City Applications//Federal University of Rio Grande do Norte Exact and Earth Sciences Center Department of Informatics and Applied Mathematics Graduate Program in Systems and Computing PhD in Computer Science. Natal-RN March 2021, P. 119 https://repositorio.ufrn.br/handle/123456789/32743.
6. Fortunato, S. (2010) Community detection in graphs. Physics Reports, Volume 486, Issues 3–5, February 2010, Pages 75-174, https://doi.org/10.1016/j.physrep.2009.11.002

7. Guerrero, C., Lera, I., & Juiz, C. (2019). A Lightweight Decentralized Service Placement Policy for Performance Optimization in Fog Computing. Journal of ambient intelligence and humanized computing. 10 – 6, pp. 2447 – 2464. SPRINGER HEIDELBERG, 01.06.2019 https://doi.org/10.48550/arXiv.2401.12699
8. Lera I., Guerrero C., Juiz C. (2019) Availability-aware Service Placement Policy in Fog Computing Based on Graph Partitions. IEEE Internet of Things Journal. 6 - 2, pp. 3641-3651. IEEE-Inst Electrical Electronics Engineers Inc, 01.04.2019 https://doi.org/10.48550/arXiv.2401.12690
9. Petrovska, I., & Kuchuk, H. (2022). Static allocation method in a cloud environment with a service model IaaS. Advanced Information Systems, 6(3), 99–106. https://doi.org/10.20998/2522-9052.2022.3.13
10. Rzepka Michał, Boryło Piotr, Marcos D. Assunção, Artur Lasoń, Laurent Lefèvre (2022) SDN-based Fog and Cloud Interplay for Stream Processing Future Generation Computer Systems/ Volume 131, June 2022, P.P. 1-17 https://doi.org/10.1016/j.future.2022.01.006
11. Salaht, S., Desprez, F., & Lebre, A. (2020). An Overview of Service Placement Problem in Fog and Edge Computing. ACM Computing Surveys, 53(3), 1-35. DOI: 10.1145/3391196
12. Singh J., Singh P., Sukhpal S. (2021) Fog computing: A taxonomy, systematic review, current trends and research challenges Journal of Parallel and Distributed Computing Volume 157, November 2021, P.P. 56-85 https://doi.org/10.1016/j.jpdc.2021.06.005
13. Syvolovskyi, I. M., Lysechko, V. P., Komar, O. M., Zhuchenko, O. S., Pastushenko, V. V.Analysis of methods for organizing distributed telecommunication systems using the paradigm of Edge Computing. 2024. *National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems*, 1(75), P. 206-211 DOI: 10.26906/SUNZ.2024.1.206.
14. Traag, V.A., Waltman, L. & van Eck, N.J. (2019) From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep 9, 5233 (2019). DOI:10.1038/s41598-019-41695-z
15. Кучук Г.А., Коваленко А.А., Лукова-Чуйко Н.В. Метод мінімізації середньої затримки пакетів у віртуальних з’єднаннях мережі підтримки хмарного сервісу. Системи управління, навігації та зв’язку. 2017. Вип. 2(42). С. 117-120.
16. Alenizi F., Rana O. (2020) Minimising Delay and Energy in Online Dynamic Fog Systems//Computer Science. Networking and Internet Architecture, https://doi.org/10.48550/arXiv.2012.12745
17. Elmokashfi A., Kvalbein A., Dovrolis C. (2010) On the Scalability of BGP: The Role of Topology Growth. IEEE Journal on Selected Areas in Communications, Vol. 28, No 8, pp. 1250-1261, October 2010. DOI:10.1109/JSAC.2010.101003.
18. Fruchterman T.M.J., Reingold E. M. (1991) Graph drawing by force-directed placement// Software: Practice and Experience, Volume 21, Issue 11, November 1991, P.P. 1129-1164 https://doi.org/10.1002/spe.4380211102
19. Reichardt J., Bornholdt S. (2004) Detecting Fuzzy Community Structures in Complex Networks with a Potts Model // APS. Physical Review Letters, 93, 218701 – Published 15 November, 2004, https://doi.org/10.1103/PhysRevLett.93.218701

Received (Надійшла) 18.03.2025
Accepted for publication (Прийнята до друку) 11.06.2025

Відомості про авторів / About the Authors

Сиволовський Ілля Михайлович – аспірант кафедри транспортного зв’язку, Український державний університет залізничного транспорту, Харків, Україна;
Syvolovskyi Illia – PhD student, Department Transport Communication, Ukraine State University of Railway Transport, Kharkiv, Ukraine;
e-mail: ilyasvl95@kart.edu.ua; ORCID Author ID: https://orcid.org/0000-0002-4592-0965.

Лисечко Володимир Петрович – доктор технічних наук, професор, начальник науково-дослідного відділу, Харківський національний університет Повітряних Сил імені Івана Кожедуба, Харків, Україна;
Lysechko Volodymyr – Doctor of Technical Sciences, Professor, Chief of the Research Department, Ivan Kozhedub Kharkiv National Air Force University, Kharkiv, Ukraine;
e-mail: lysechkov@ukr.net; ORCID Author ID: https://orcid.org/0000-0002-1520-9515;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57195515166

Метод ієрархічної кластеризації вузлів розподілених телекомунікаційних систем з використанням графових алгоритмів

І. М. Сиволовський, В. П. Лисечко

Анотація. У статті розроблено модифікований метод ієрархічної кластеризації обчислювальних вузлів у розподілених телекомунікаційних системах із врахуванням продуктивності вузлів, топології мережі, затримок та пропускної здатності каналів зв’язку. Запропонований метод ґрунтується на модифікованому алгоритмі Louvain, що виконує багатокрокову кластеризацію графа з динамічним коригуванням параметрів. Це дозволяє контролювати розмір кластерів та їхню внутрішню щільність, відповідно до заданих цільових показників, мінімізуючи фрагментацію мережевої структури. На основі порівняльного аналізу сучасних методів кластеризації та проведеного експериментального моделювання встановлено, що запропонований метод забезпечує зменшення фрагментації кластерів на 36% порівняно з методом Leiden. Крім того, він дозволяє знизити міжкластерні затримки на 4,5% у порівнянні з методом Louvain та на 11,8% порівняно з Leiden, що свідчить про ефективнішу організацію міжкластерної взаємодії. Вдосконалений метод забезпечує рівномірний розподіл обчислювальних вузлів між кластерами, що сприяє оптимізації загальної продуктивності розподіленої телекомунікаційної системи.

Ключові слова: розподілені телекомунікаційні системи, ієрархічна кластеризація вузлів, моделювання інтелектуальних систем, графова алгоритмічна оптимізація, Louvain / Leiden Clustering, балансування навантаження, пропускна здатність, оптимізація топології, мінімізація затримок.