Graph Partitioning Algorithms: A Comparative Study

Rafael M. S. Siqueira, Alexandre D. Alves, Otávio A. O. Carpinteiro, and Edmilson M. Moreira

Abstract

One of the classic problems related to graphs is partitioning their vertices into subsets, consisting of composing groups with high connectivity. The graph partitioning problem is of interest since the amount of data generated today is gigantic, and the importance of determining groups is essential for making strategic decisions in several areas. This paper compares the main graph partitioning methods found in the literature, considering the minimum cut criteria and load balancing factors in different types of graphs.

Keywords

Graph theory · Graph algorithms · Graph partitioning · Graph clustering · Minimum cuts · Load balancing factor · Heuristic methods · Agglomerative methods · Multilevel methods · Spectral methods

1 Introduction

A graph is a pair of sets (V, E), where V is the set of vertices and E is the set of edges formed by pairs of vertices. The graph's vertices can represent many entities in the real world, relating to each other for different reasons, conditions, or behaviors. Thus, a group of vertices connected by a dense set of edges represents behavior that contains a greater affinity.

R. M. S. Siqueira · O. A. O. Carpinteiro · E. M. Moreira (✉)
Federal University of Itajubá, Itajubá, MG, Brazil
e-mail: edmarmo@unifei.edu.br

A. D. Alves
Federal University of ABC, Santo André, SP, Brazil
e-mail: alexandre.donizeti@ufabc.edu.br

One of the classic problems related to graphs is partitioning their vertices into subsets, consisting of composing groups with high connectivity. We found several nomenclatures for this situation, such as the clustering problem or detection of communities. The graph partitioning problem is of interest since the amount of data generated today is gigantic, and determining groups is essential for making strategic decisions in several fields, such as complex networks [17], clustering [6], and operational research [16].

The most common form of computational representation of graphs is through structures such as matrices and lists. However, the current volume of information has become a limiting factor for representations in main memory. Because of that came the graph-oriented databases based on the NoSQL paradigm [18, 20]. Several approaches define partitioning algorithms' behavior, each with its primary objective, advantages, and situations that present themselves as the most efficient way. This paper compares the main methods found in the literature, adopting an implementation approach based on disk persistence with technology using a graph-oriented database (Neo4j).

2 Partitioning Methods

Graph partitioning has been studied for many decades and is considered one of the mathematical problems of great interest [4]. A vertex in the graph can represent an element or individual, while an edge between two vertices indicates a relationship between these objects. Moreover, some of these vertices have similarities, affinity, or similar roles in a graph or network. Thus, there are possibilities to group these vertices to determine sets that influence or characterize the graph. Determining partitions in an arbitrary graph is not a computationally easy task, especially when a graph has many vertices. Usually, the number of partitions is unknown, and

S. Latifi (ed.), *ITNG 2024: 21st International Conference on Information Technology-New Generations*, Advances in Intelligent Systems and Computing 1456,
https://doi.org/10.1007/978-3-031-56599-1_65

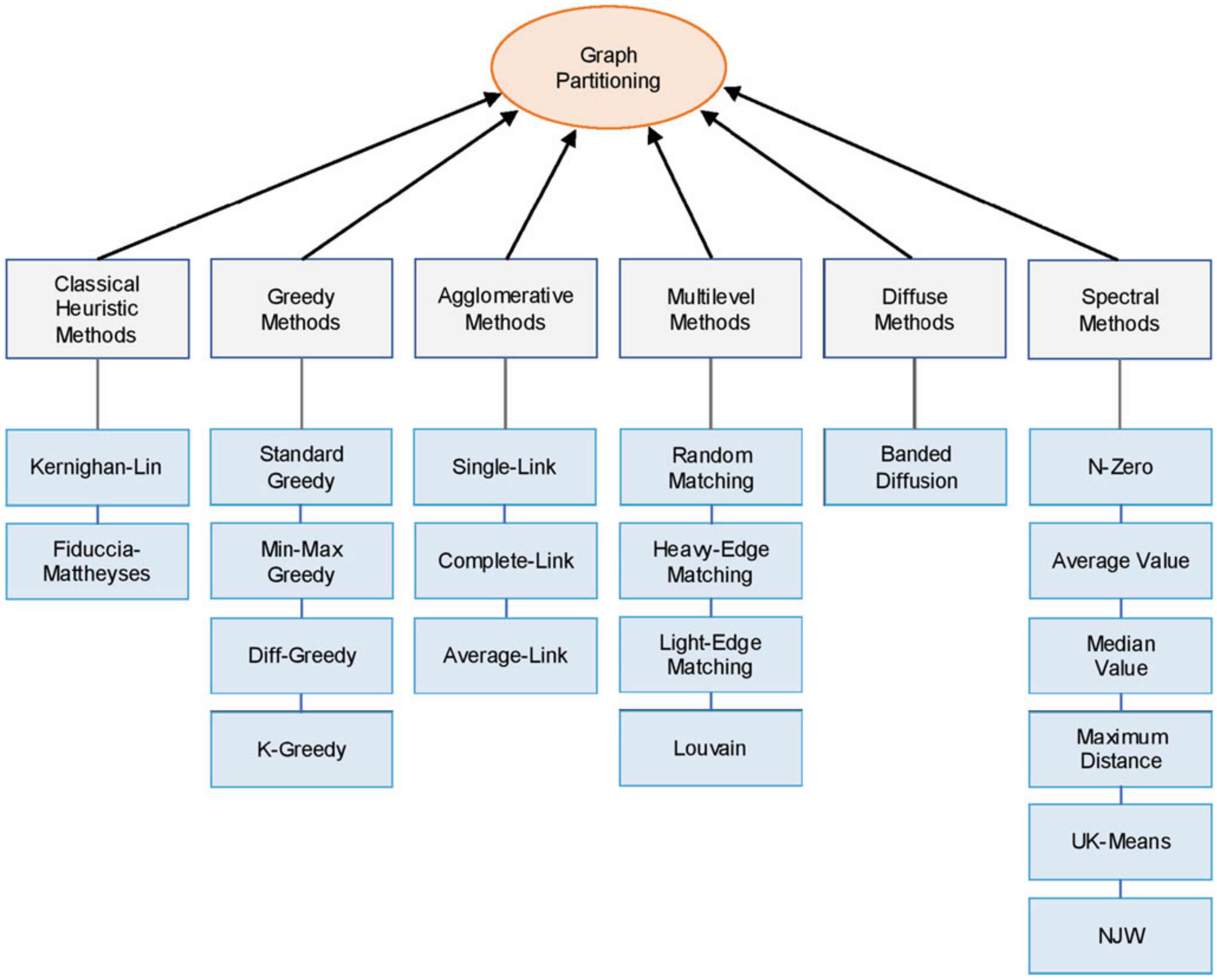

Fig. 1 Structure of the graph partitioning algorithms

their sizes and densities are different. As a result, several methods were developed, each with varying levels of success and emphasizing specific objectives.

In this work, we analyzed the most common methods found in the literature, allowing us to compare their executions and determine their efficiency regarding the quality of the resulting partitions. Figure 1 shows the division of these algorithms.

Heuristic methods are exploratory algorithms that try to approximate their resolution to the optimal result but enabling their application in practice. These methods tend to find the best possible solutions rather than exact solutions. For the classical heuristic method, we considered the Kernighan-Lin [15] and Fiduccia-Mattheyses [8] algorithms.

Greedy methods are heuristic methods that follow a basic strategy of applying an action that moves toward the solution at each step. In these methods, vertices without partitions are inserted alternately into each partition so that the new vertex added results in a better cut value for the graph. The algorithms used in the analysis were Standard Greedy [7], Min-Max Greedy [2], Diff-Greedy [1], and K-Greedy [11].

Agglomerative methods create a hierarchy of relationships between the elements or between the vertices of the graph [9]. The algorithms of agglomerative methods analyzed were Single-link [23], Complete-link [10], and Average-link [13].

Multilevel methods [14] consist of grouping edges or vertices at each graph representation level, considerably reducing their size, determining the partitions in their minimum representation, propagating the result in the original graph, and defining its partitions. The multilevel methods are represented by Random Matching [12], Heavy-Edge Matching, Light-Edge Matching, and Louvain [3].

Pellegrini [21] proposed a global diffusion integration model compared to a system of tanks and pipes. The diffuse method is represented by the main algorithm of this class named Banded Diffusion.

The concept of Spectral Graph Theory is the basis of spectral graph partitioning methods. Thus, the eigenvalues, eigenvectors, and Laplacian matrix are considered. In the spectral method were considered N-Zero [22], Average Value [5], Median Value [24], Maximum Distance [22], UK-Means [25], and NJW [19].

3 Experiments

We performed the tests on a computer with an *Intel Core i7-4500U CPU 1.80GHz*, 8 GB of memory *DDR3L MHz* SDRAM and a Linux operating system in the *Ubuntu 16.04.1 LTS x64* distribution. The database is the *Neo4j 3.0 Community*, and the implementation of the algorithms was performed on the *Java SE platform (build 1.8.0_92-b14)*.

We used 60 graph instances (6 different sizes) with their random composition for the computational experiments. We segmented them into two balanced partitions concerning the number of vertices and the lowest value for the minimum cut. All graphs are connected with weighted and non-directed edges. Each group of graphs of the same size receives a specific name. Thus, graphs of 100 vertices are called *G1*, and graphs of 200, 300, 400, 500, and 1000 are called *G2*, *G3*, *G4*, *G5*, and *G6*, respectively. All instances consist of random and synthetic artificial graphs with variations in the formation of edge arrangements; composing structures are known in the literature as *cluster*, *flower*, *grid*, and *lobster*.

4 Results and Discussions

The methodology adopted consisted of applying each algorithm in the available instances to compare the performance and quality of the obtained partitions.

Each of the algorithms was applied ten times on the same instance (without considering the warm-up execution), performing the average calculation of the results of each evaluation parameter to determine the value to be compared between the methods. Thus, the results were approximated for the algorithms with random components, while exact solutions presented precise results.

The performance analysis of the algorithms is divided into two parts, each referring to the measurement parameter of the quality of the result obtained (minimum cut and load balancing factor). Thus, the analysis is performed for the different arrangements presented (cluster, flower, grid, and lobster). The results compared refer to each of the instances of experimentation (*G1*, *G2*, *G3*, *G4*, *G5*, and *G6*). For each of them, ten graphs are generated randomly, according to the description above.

4.1 Minimum Cuts

The minimum cut consists of finding the minimum value in the sum of the edge weights related to different partitions' vertices. Table 1 shows results obtained for the cluster graphs.

The Kernighan-Lin and Fiduccia-Mattheyses algorithms present linear behavior concerning the number of vertices of the graph. This behavior is because these solutions seek a balance between the number of vertices of the resulting partitions and not the minimum cut. However, one notices slightly better results for the Fiduccia-Mattheyses algorithm.

Among the greedy methods, the Min-Max algorithm stands out, with results on average three times better than the others. Agglomerative algorithms obtain the best results. However, to achieve these results, the algorithms generate completely unbalanced partitions.

The Banded Diffusion algorithm performs slightly better than Min-Max Greedy. However, its random component of the initial choice of vertices can directly affect the results, increasing the instability of the results in cluster graphs with a large number of vertices. In the spectral algorithms, it is possible to notice a big difference between them because the minimum cut of the N-Zero and Mean Value algorithms is close to the classical heuristics algorithms. The UK-Means algorithm behaves similarly to multilevel and greedy algorithms in graphs G1, G2, G3, and G4. However, the influence

Table 1 Minimum cut found in cluster graphs

Method/Algorithm	G1	G2	G3	G4	G5	G6
Classical heuristic						
Kernighan-Lin	256	539	747	1010	1228	2552
Fiduccia-Mattheyses	195	363	538	699	1001	2099
Greedy						
Standard Greedy	119	293	294	484	439	1687
Min-Max Greedy	101	256	242	246	349	1169
Diff-Greedy	117	257	217	364	482	1123
K-Greedy	120	294	291	390	449	1458
Agglomerative						
Single-Link	3	8	3	4	3	13
Complete-Link	12	8	11	13	4	7
Average-Link	3	8	11	7	3	9
Multilevel						
Random Matching	54	157	412	449	477	1616
Heavy-Edge Matching	64	175	253	568	463	1326
Light-Edge Matching	61	216	279	518	422	1442
Louvain	81	157	442	415	468	1660
Diffuse						
Banded Diffusion	1	12	95	168	241	337
Spectral						
N-Zero	246	486	717	962	1252	2486
Average Value	246	486	717	962	1252	2486
Median Value	13	9	12	16	11	20
Maximum Distance	20	24	17	19	21	20
UK-Means	166	407	432	931	336	1140
NJW	117	198	246	499	286	1421

Table 2 Minimum cut found in flower graphs

Method/Algorithm	G1	G2	G3	G4	G5	G6
Classical heuristic						
Kernighan-Lin	53	103	153	203	253	503
Fiduccia-Mattheyses	49	95	132	189	217	439
Greedy						
Standard Greedy	16	21	31	35	46	37
Min-Max Greedy	11	11	13	16	22	23
Diff-Greedy	12	14	19	20	25	24
K-Greedy	13	18	25	18	26	19
Agglomerative						
Single-Link	3	3	3	3	3	3
Complete-Link	3	3	3	3	3	3
Average-Link	3	3	3	3	3	3
Multilevel						
Random Matching	14	25	25	25	33	45
Heavy-Edge Matching	14	13	15	15	18	18
Light-Edge Matching	14	19	18	18	15	16
Louvain	17	38	39	43	61	75
Diffuse						
Banded Diffusion	7	7	6	7	7	7
Spectral						
N-Zero	71	159	232	286	381	765
Average Value	71	159	232	286	381	765
Median Value	6	10	6	10	6	8
Maximum Distance	5	9	6	4	10	11
UK-Means	75	123	220	257	335	379
NJW	77	73	115	159	161	194

Table 3 Minimum cut found in grid graphs

Method/Algorithm	G1	G2	G3	G4	G5	G6
Classical heuristic						
Kernighan-Lin	146	289	430	594	719	1434
Fiduccia-Mattheyses	66	124	178	245	291	573
Greedy						
Standard Greedy	26	40	54	74	77	118
Min-Max Greedy	22	34	41	58	65	73
Diff-Greedy	22	30	51	65	72	102
K-Greedy	23	43	52	70	87	117
Agglomerative						
Single-Link	4	4	4	4	4	3
Complete-Link	4	4	4	4	3	4
Average-Link	4	4	3	4	4	4
Multilevel						
Random Matching	15	29	23	41	45	50
Heavy-Edge Matching	13	26	31	41	47	66
Light-Edge Matching	17	31	28	40	48	68
Louvain	16	27	26	34	44	43
Diffuse						
Banded Diffusion	13	17	22	26	27	40
Spectral						
N-Zero	106	186	293	405	502	967
Average Value	106	186	293	405	502	967
Median Value	18	25	34	36	39	57
Maximum Distance	4	2	1	7	1	1
UK-Means	96	73	186	166	352	676
NJW	54	76	93	103	208	262

of the initial random determination causes its instability to be noticed in G5 and especially in G6.

The behavior of the algorithms can change depending on the graph to which the partitioning is being applied. Therefore, it is important to use the algorithms in other models. In this way, the partitioning models were applied in a set of graphs that present the *flower* model. This model has a well-determined geometric positioning of vertices and almost no construction randomness influencing the results. Therefore, it is possible to determine the behavior of the algorithms in cases where the vertices have approximately the same degree and there is only one edge removed to indicate the natural cut-off point. The results obtained for the minimum cut for the flower model can be seen in Table 2.

It is possible to notice two types of behavior. The first produces the expected result. The minimum cut weight changes little as the algorithm is applied to larger graphs because the ideal cutting point has this characteristic when combined with the geometry of the flower model. The second type of behavior is just the opposite. As the number of vertices of the graph increases, the weight of the minimum cut also increases because the algorithm cannot detect the geometry or the natural cut region, segmenting the graph as a whole into rings and putting; as a result, one or more complete rings in a partition.

The Kernighan-Lin and Fiduccia-Mattheyses algorithms follow the second pattern of behavior described. The same happens for the N-Zero, Average Value, UK-Means, and NJW algorithms. The greedy algorithms present better results than those already mentioned, with acceptable behaviors, but not the best results for this model. Agglomerative algorithms also obtain the best results for the minimum cut. They produce entirely unbalanced partitions, with an average of 3 vertices in one partition, while the rest compose the other partition.

Multilevel algorithms achieve good results with relative values for different graph sizes. The Louvain algorithm presented a slightly worse outcome due to the instability of the results in larger graphs, presenting difficulties in detecting the geometry of the graphs. The Banded Diffusion algorithm presented stable results, following an operational logic capable of adapting to the graph's geometry. The Maximum Distance and Median Value spectral algorithms present the best results for their group. Both can detect the equal influence of the vertices and detect the natural cut-off point.

The grid model is a variation that is very close to the flower model but with some characteristics that make it even more challenging to determine the ideal cutoff point since the ideal weight is not constant. The results obtained for the minimum cut weight, in this case, can be seen in Table 3.

The incremental behavior of the minimum cut weight is perceived in higher-order graphs. Classical heuristics algorithms do not get good results, as they focus on improving the balance at the expense of the minimum cut weight. The greedy algorithms continue with the same performances noted in the experiment with flower-type arrays. Likewise, the agglomerative algorithms remain, resulting in low minimum cuts. However, it is possible to notice that their values do not follow the increment of the ideal border region of the graph instances. These results are obtained from a partition with few vertices and the other with most graph vertices. The multilevel algorithms obtained excellent results because they always operate by efficient criteria for choosing vertices to incorporate the partition of this vertex clustering model.

As in the experiment with flower graphs, the Banded Diffusion algorithm got one of the best results, demonstrating that it adapts well to the geometric dynamics of the graph instances and can detect naturally ideal slice regions.

The spectral algorithms showed some instability. In the UK-Means and NJW algorithms, it is possible to notice variations in the results of the minimum cut due to the choice of random seed in the determination of centroids. The Maximum Distance algorithm was the most affected by how the vertices are related in this graph model. In some cases, the partitioning was impossible, and in others, the partitions were quite unbalanced, resulting in imprecise minimum cuts. The Median Value spectral algorithm stands out for adapting well to the changing dynamics of the relationship between the vertices and presenting good results in the minimum cut, very close to the results obtained by the Banded Diffusion algorithm.

The lobster type has a peculiar connection between the vertices, generating little dense graphs with an average degree between 2 and 3. This structure makes it even more challenging to determine the partitions, impacting the algorithms that use random seeds.

The results of the minimum cuts for applications on instances of graphs with relationship between vertices of lobster type can be analyzed according to Table 4. The results show that the classical heuristics algorithms are significantly affected by the way they start since the partitions are randomly determined as a starting point. Therefore, the problem only gets worse as the partitioned graph has a more significant number of vertices.

Greedy algorithms also yield hampered results by randomly determining their starting points. This impact occurs on a smaller scale because there is a systematization for the partitioning process. Apparently, agglomerative algorithms achieve good results, but again it is since most of the time, one of the partitions has only a single vertex. Multilevel algorithms produce the best results, as they can adapt to the conditions of the partitioned graph and achieve ideal results with unitary cutoff values. The Banded Diffusion algorithm often achieves excellent minimum cut results, but its random starting component influences the overall result, directly impacting the quality of the partitions.

Table 4 Minimum cut found in lobster graphs

Method/Algorithm	G1	G2	G3	G4	G5	G6
Classical heuristic						
Kernighan-Lin	43	93	161	209	250	526
Fiduccia-Mattheyses	35	69	101	135	153	323
Greedy						
Standard Greedy	12	16	30	31	30	71
Min-Max Greedy	6	11	10	12	10	17
Diff-Greedy	6	8	13	11	11	15
K-Greedy	4	4	6	7	5	17
Agglomerative						
Single-Link	1	5	8	1	1	1
Complete-Link	3	3	7	1	1	1
Average-Link	2	3	5	1	1	1
Multilevel						
Random Matching	1	1	1	1	1	1
Heavy-Edge Matching	1	1	1	1	1	1
Light-Edge Matching	1	1	1	1	1	1
Louvain	1	1	1	1	1	1
Diffuse						
Banded Diffusion	1	1	3	2	1	1
Spectral						
N-Zero	51	103	151	201	256	496
Average Value	51	103	151	201	256	496
Median Value	40	82	118	157	212	366
Maximum Distance	2	1	22	31	30	32
UK-Means	51	82	47	53	89	186
NJW	33	33	19	35	49	9

Again, the Average Value and N-Zero spectral algorithms present the same results because they share similar techniques and cannot follow the graph dynamics, which also occurs with the Median Value spectral algorithm. The Maximum Distance spectral algorithm presents less instability in the results in small graphs. In larger graphs, this instability increases when calculating the minimum cut.

4.2 Load Balancing Factor

The load balancing factor is one of the main criteria for evaluating the quality of graph partitioning. In addition to minimizing clipping, a good algorithm should obtain partitions with approximately the same number of vertices. The first experiments were performed on cluster graphs, resulting in the load balancing factors presented in Table 5. The values are represented by a number between zero and one, where the ideal value is unity.

Classical heuristic algorithms operate with a focus on obtaining better results concerning the load balancing factor. The Kernighan-Lin algorithm achieved the ideal result in all instances, while the Fidducia-Mattheyses algorithm achieved

Table 5 Load balancing in cluster graphs

Method/Algorithm	G1	G2	G3	G4	G5	G6
Classical heuristic						
Kernighan-Lin	1.00	1.00	1.00	1.00	1.00	1.00
Fiduccia-Mattheyses	0.95	0.95	0.93	0.50	0.60	0.80
Greedy						
Standard Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Min-Max Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Diff Greedy	1.00	1.00	1.00	1.00	1.00	1.00
K-Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Agglomerative						
Single-Link	0.02	0.01	0.01	0.01	0.00	0.00
Complete-Link	0.02	0.01	0.01	0.01	0.00	0.00
Average-Link	0.02	0.01	0.01	0.01	0.20	0.00
Multilevel						
Random Matching	0.74	0.60	0.77	0.60	0.55	0.65
Heavy-Edge Matching	0.72	0.59	0.68	0.76	0.80	0.90
Light-Edge Matching	0.74	0.65	0.50	0.67	0.68	0.78
Louvain	0.44	0.52	0.66	0.56	0.54	0.67
Diffuse						
Banded Diffusion	1.00	0.99	0.79	0.73	0.59	0.65
Spectral						
N-Zero	0.96	0.93	0.99	0.91	0.99	0.97
Average Value	0.96	0.93	0.99	0.91	0.99	0.97
Median Value	0.98	0.99	0.99	1.00	0.07	0.07
Maximum Distance	0.04	0.02	0.04	0.01	0.01	0.00
UK-Means	0.45	0.72	0.96	0.67	0.07	0.00
NJW	0.81	0.58	0.41	0.77	0.44	0.74

Table 6 Load balancing in flower graphs

Method/Algorithm	G1	G2	G3	G4	G5	G6
Classical heuristic						
Kernighan-Lin	1.00	1.00	1.00	1.00	1.00	1.00
Fiduccia-Mattheyses	0.89	0.94	0.96	0.96	0.96	0.96
Greedy						
Standard Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Min-Max Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Diff Greedy	1.00	1.00	1.00	1.00	1.00	1.00
K-Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Agglomerative						
Single-Link	0.02	0.06	0.01	0.01	0.00	0.00
Complete-Link	0.02	0.06	0.01	0.01	0.00	0.00
Average-Link	0.02	0.06	0.01	0.01	0.00	0.00
Multilevel						
Random Matching	0.45	0.72	0.40	0.63	0.27	0.34
Heavy-Edge Matching	0.63	0.61	0.67	0.73	0.53	0.52
Light-Edge Matching	0.74	0.73	0.80	0.61	0.58	0.78
Louvain	0.69	0.53	0.46	0.57	0.56	0.53
Diffuse						
Banded Diffusion	0.91	0.98	0.99	0.99	0.99	1.00
Spectral						
N-Zero	0.64	0.93	0.95	0.95	0.96	0.99
Average Value	0.64	0.93	0.95	0.95	0.96	0.99
Median Value	0.96	0.98	0.99	0.99	0.99	1.00
Maximum Distance	0.06	0.05	0.05	0.01	0.06	0.09
UK-Means	0.70	0.52	0.86	0.60	0.70	0.59
NJW	0.68	0.62	0.66	0.79	0.52	0.65

good results on the smallest instances. The difference between the algorithms is due to the improvement in the minimum cut weight of the second in relation to the first.

Greedy algorithms also achieve optimal balance. At each iteration of these algorithms, a partition receives the increment of one more vertex. Thus, all partitions end with the same number of vertices. Agglomerative algorithms have the worst results. This fact is due to the behavior already described. One of the partitions receives an average of three vertices, while the other has the other vertices. Multilevel algorithms present varying results. There is a performance variation in the partitioning processes, as they start their processes with the choice of initial random seeds in unfavorable positions, influencing the results in general. The Banded Diffusion algorithm showed better results for the smallest graphs. Sometimes, randomly chosen initial vertices are too close together and should belong to the same partition, affecting the balance quality.

The results obtained for the N-Zero, Average Value, and Median Value spectral algorithms are very close to ideal. However, the Maximum Distance, NJW, and UK-Means algorithms suffer from the same instabilities mentioned before, in which the determination of random seeds impacts the results.

It is noticeable that some methods have more efficient algorithms for the load balancing factor between partitions, regardless of the minimum cutoff values presented before. It is noticed that it is often necessary to reduce the quality of results from one parameter to achieve greater efficiency in another (as explained in classical heuristics algorithms). The results for the flower graphs are shown in Table 6.

The greedy algorithms remain with ideal results. The results are stable, constant, and independent of the geometry of the partitioned graph. The Banded Diffusion algorithm achieved a result close to the ideal value. Agglomerative algorithms continue with the worst results, while multilevel algorithms vary significantly in their final values. As with the minimum cut, these algorithms partition instances into rings and place them into the same partition. Usually, these rings are divided in approximately 1/3 or 2/3, making the factor fall between 0.30 and 0.70.

In most cases, the Mean Value, N-Zero, and Median Value spectral algorithms present close to ideal results. The Maximum Distance algorithm does not achieve good results, as its behavior for flower graphs is similar to agglomerative algorithms, with completely unbalanced partitions. In general, the results were not significantly different from the previous ones, regardless of the application to instances with clusters or flowers graphs. The results obtained can be seen in Table 7.

The results of this type of graph show a slight variation concerning the values obtained in flower graphs. Classical,

Table 7 Load balancing in grid graphs

Method/Algorithm	G1	G2	G3	G4	G5	G6
Classical heuristic						
Kernighan-Lin	1.00	1.00	1.00	1.00	1.00	1.00
Fiduccia-Mattheyses	0.81	0.96	0.93	0.97	0.97	0.98
Greedy						
Standard Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Min-Max Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Diff Greedy	1.00	1.00	1.00	1.00	1.00	1.00
K-Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Agglomerative						
Single-Link	0.02	0.06	0.01	0.01	0.00	0.00
Complete-Link	0.02	0.06	0.01	0.01	0.00	0.00
Average-Link	0.02	0.06	0.01	0.01	0.00	0.00
Multilevel						
Random Matching	0.28	0.42	0.24	0.42	0.41	0.36
Heavy-Edge Matching	0.44	0.48	0.49	0.53	0.56	0.66
Light-Edge Matching	0.50	0.58	0.54	0.45	0.48	0.61
Louvain	0.29	0.47	0.36	0.42	0.40	0.29
Diffuse						
Banded Diffusion	0.64	0.74	0.69	0.88	0.76	0.83
Spectral						
N-Zero	0.98	0.98	0.86	0.90	0.92	0.98
Average Value	0.98	0.98	0.86	0.90	0.92	0.98
Median Value	0.98	0.98	1.00	0.92	0.96	0.99
Maximum Distance	0.02	0.01	0.01	0.07	0.02	0.00
UK-Means	0.87	0.07	0.45	0.21	0.56	0.70
NJW	0.85	0.57	0.44	0.08	0.57	0.33

Table 8 Load balancing in lobster graphs

Method/Algorithm	G1	G2	G3	G4	G5	G6
Classical heuristic						
Kernighan-Lin	1.00	1.00	1.00	1.00	1.00	1.00
Fiduccia-Mattheyses	0.90	0.96	0.94	0.98	0.96	0.98
Greedy						
Standard Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Min-Max Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Diff Greedy	1.00	1.00	1.00	1.00	1.00	1.00
K-Greedy	1.00	1.00	1.00	1.00	1.00	1.00
Agglomerative						
Single-Link	0.02	0.06	0.03	0.00	0.00	0.00
Complete-Link	0.02	0.06	0.01	0.00	0.00	0.00
Average-Link	0.02	0.06	0.01	0.00	0.00	0.00
Multilevel						
Random Matching	0.03	0.11	0.07	0.02	0.02	0.01
Heavy-Edge Matching	0.07	0.04	0.12	0.03	0.01	0.01
Light-Edge Matching	0.06	0.06	0.11	0.03	0.01	0.01
Louvain	0.11	0.12	0.15	0.71	0.91	0.89
Diffuse						
Banded Diffusion	0.49	0.61	0.50	0.51	0.79	0.82
Spectral						
N-Zero	0.94	0.90	0.97	0.97	0.92	0.98
Average Value	0.94	0.90	0.97	0.97	0.92	0.98
Median Value	0.98	0.99	0.99	1.00	1.00	1.00
Maximum Distance	0.02	0.01	0.01	0.02	0.00	0.00
UK-Means	0.52	0.26	0.05	0.02	0.06	0.36
NJW	0.65	0.40	0.05	0.03	0.15	0.01

greedy, and agglomerative heuristics algorithms maintained practically the same performance. The same happened with the multilevel and spectral algorithms, keeping their values and stability variations. The only algorithm that behaved differently was Banded Diffusion. This algorithm works with a random seed that determines the starting point of the partitions. In flower graphs, the ideal choice did not always happen, impacting the final value obtained.

Experiments were carried out with lobster graphs as the last context for applying partitioning algorithms to synthetic graphs. This type of graph establishes a large variation between vertex distances and affects algorithms that contain a random component. The best results usually start with choosing the most distant vertices as a starting point for each partition. Algorithms that make decisions after determining starting points are also affected but to a lesser extent. The results for this graphs are shown in Table 8.

Classical and greedy heuristics algorithms continue to find ideal results. In turn, the Light-Edge Matching, Heavy-Edge Matching, and Random Pairing multilevel algorithms had poor results, indicating a lot of difficulty in determining contraction points in higher-level constructions. The Louvain algorithm presents problems in determining good partitions in small graphs. The algorithm performs contractions with neighboring vertices based on tests to improve the overall modularity gain of the graph. The more longitudinal the graph is, the better the results are. Thus, the performance is low in graphs with few vertices, significantly improving the results in the G5 and G6 configuration graphs.

The performance of the Banded Diffusion algorithm is also affected, mainly in smaller graphs, since the probability of determining the starting points of each partition randomly very close is more significant for graphs of these sizes. The Mean Value, N-Zero, and Median Value spectral algorithms continue with excellent results. On the other hand, the NJW and UK-Means algorithms present very unstable results, resulting from the random determinations of the starting point of their partitioning processes.

The Maximum Distance algorithm has such difficulty partitioning this type of graph that the load balancing factor for instances with the highest number of vertices is practically null. This indicates that one of the partitions often has only one vertex, while the rest make up the other partition (as well as agglomerative algorithms).

5 Conclusions

This article presented a comparative study on the problem of graph partitioning and the most common solutions found in the literature, with an analysis of two evaluation criteria: minimum cut and load balancing factor. The most noticeable result in the experiments was the imbalance in the number of vertices between partitions produced by some algorithms. The agglomerative algorithms generated the most evident distortions since, in all the cases tested, the result was a partition with a maximum of 5 vertices. This effect occurs in cases where the choice of vertices to be aggregated in the partition ends up selecting the same group already incremented several times. Thus, the group of vertices grows disproportionately, unbalancing the load balancing factor. The best minimum cut results were established by Banded Diffusion and Median Value spectral algorithms. The minimum cut-off value for both in clustered graphs is excellent, presenting a constancy of results in smaller graphs. On the other hand, in graphs with a larger number of vertices, the result obtained from the Banded Diffusion algorithm is kept at the cost of the balancing factor.

References

1. R. Battiti, A. Bertossi, Differential greedy for the 0-1 equicut problem, in *Proceedings of the DIMACS Workshop on Network Design: Connectivity and Facilities Location* (1997) , pp. 3–21
2. R. Battiti, A.A. Bertossi, Greedy, prohibition, and reactive heuristics for graph partitioning. IEEE Trans. Comput. **48**(4), 361–385 (1999)
3. V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks. J. Stat. Mech. Theory Experiment **2008**(10), P10008 (2008)
4. Ü. Çatalyürek, K. Devine, M. Faraj, L. Gottesbüren, T. Heuer, H. Meyerhenke, P. Sanders, S. Schlag, C. Schulz, D. Seemaier et al., More recent advances in (hyper) graph partitioning. ACM Comput. Surv. **55**(12), 1–38 (2023)
5. T.F. Chan, P. Ciarlet Jr., W. Szeto, On the optimality of the median cut spectral bisection graph partitioning method. SIAM J. Sci. Comput. **18**(3), 943–948 (1997)
6. I.A. Ebeid, J.R. Talburt, M.A.S. Siddique, Graph-based hierarchical record clustering for unsupervised entity resolution, in 19th ITNG (2022), pp. 107–118
7. J. Edmonds, Matroids and the greedy algorithm. Math. Program. **1**(1), 127–136 (1971)
8. C.M. Fiduccia, R.M. Mattheyses, A linear-time heuristic for improving network partitions, in *Design Automation, 1982. 19th Conference on* (IEEE, 1982), pp. 175–181
9. M. Hajiaghayi, T. Johnson, M.R. Khani, B. Saha, Hierarchical graph partitioning, in *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures* (2014), pp. 51–60
10. P. Hansen, M. Delattre, Complete-link cluster analysis by graph coloring. J. Am. Stat. Assoc. **73**(362), 397–403 (1978)
11. D. Hausmann, B. Korte, K-greedy algorithms for independence systems. Z. Oper. Res. **22**(1), 219–228 (1978)
12. B. Hendrickson, R.W. Leland, A multi-level algorithm for partitioning graphs. SC **95**, 28 (1995)
13. A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data* (Prentice-Hall, 1988)
14. G. Karypis, V. Kumar, Multilevel graph partitioning schemes, in *ICPP (3)* (1995), pp. 113–122
15. B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**(2), 291–307 (1970)
16. A.A. Khan, M.U. Khan, M. Iqbal, Multilevel graph partitioning scheme to solve traveling salesman problem, in *9th ITNG* (2012), pp. 458–463
17. N. Meghanathan, Complex network analysis of the us marine highway network, in *20th ITNG* (2023) , pp. 437–443
18. L.A. Neumann, E. Seraphim, O.A. Carpinteiro, E.M. Moreira, Participatory modeling: A new approach to model graph-oriented databases, in *19th ITNG* (2022), pp. 97–106
19. A.Y. Ng, M.I. Jordan, Y. Weiss et al., On spectral clustering: Analysis and an algorithm. Adv. Neural Inf. Process. Syst. **2**, 849–856 (2002)
20. A.T. Oliveira, A.D. Souza, E.M. Moreira, E. Seraphim, Mapping and conversion between relational and graph databases models: A systematic literature review, in *17th ITNG* (2020), pp. 539–543
21. F. Pellegrini, A parallelisable multi-level banded diffusion scheme for computing balanced partitions with smooth boundaries, in *European Conference on Parallel Processing* (Springer, 2007), pp. 195–204
22. A. Pothen, H.D. Simon, K.P. Liou, Partitioning sparse matrices with eigenvectors of graphs. SIAM J. Matrix Anal. Appl. **11**(3), 430–452 (1990)
23. R. Sibson, Slink: an optimally efficient algorithm for the single-link cluster method. Comput. J. **16**(1), 30–34 (1973)
24. D.A. Spielmat, S.H. Teng, Spectral partitioning works: Planar graphs and finite element meshes, in *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on* (1996), pp. 96–105
25. U. Von Luxburg, A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)