J Braz Comput Soc (2011) 17: 19–29
DOI 10.1007/s13173-010-0027-x

ORIGINAL PAPER

A graph clustering algorithm based on a clustering coefficient for weighted graphs

Mariá C.V. Nascimento · André C.P.L.F. Carvalho

Received: 27 May 2010 / Accepted: 29 November 2010 / Published online: 21 December 2010

Abstract Graph clustering is an important issue for several applications associated with data analysis in graphs. However, the discovery of groups of highly connected nodes that can represent clusters is not an easy task. Many assumptions like the number of clusters and if the clusters are or not balanced, may need to be made before the application of a clustering algorithm. Moreover, without previous information regarding data label, there is no guarantee that the partition found by a clustering algorithm automatically extracts the relevant information present in the data. This paper proposes a new graph clustering algorithm that automatically defines the number of clusters based on a clustering tendency connectivity-based validation measure, also proposed in the paper. According to the computational results, the new algorithm is able to efficiently find graph clustering partitions for complete graphs.

Keywords Clustering coefficient · Graph clustering · Combinatorial optimization

1 Introduction

Data clustering deals with the discovery of data patterns, in the form of data clusters, in the objects from a dataset. For such, objects with similar characteristics are placed into the same group (cluster) and objects with different features are placed into different clusters. Several clustering algorithms have been proposed in the literature, based on several approaches. These algorithms have been successfully applied to a wide variety of problems, including applications from areas like bioinformatics [11], image processing [25] and market segmentation [2]. Despite the good results obtained by the use of clustering algorithms in several problem domains, cluster analysis is still seen as a challenging problem. Some particular clustering problems require more sophisticated algorithms.

A specific problem of clustering is known as graph clustering. Graph clustering looks for patterns among nodes in a graph in order to produce a meaningful node partitioning. Many inferences about the node partition may provide useful information regarding the data. A few examples of the benefits, as well as different approaches for graph clustering, can be found in [24].

Additionally, in spite of the large number of clustering algorithms, few of them are able to automatically discover partitions without the information of the number of clusters beforehand. Automatic graph clustering algorithms, able to define by themselves the number of clusters, play an important role in data analysis, since they allow a more efficient application of clustering algorithms to a dataset without prior knowledge of the data conformation. Therefore, the investigation of new clustering algorithms able to deal with graph clustering problems and to automatically define the number of clusters is an important research issue.

Automatic clustering algorithms usually rely on a validation criterion to select the number of clusters. Moreover, the evaluation of the quality of clustering partitions is not as simple and direct as the evaluation of classification models. Several validation measures have been specifically designed to assess graph clustering partitions. Among them, we can cite modularity [19]. Modularity evaluates the difference be-

M.C.V. Nascimento (✉) · A.C.P.L.F. Carvalho
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, Caixa Postal 668, São Carlos, SP,
CEP 13560-970, Brazil
e-mail: mariah@icmc.usp.br

A.C.P.L.F. Carvalho
e-mail: andre@icmc.usp.br

tween the presence of an edge connecting a pair of nodes from a same cluster and the probability that the same edge can be found at the same cluster in a random graph. The lower this probability, the higher the measure value. Thus, the best value is the highest value that can be achieved.

A measure frequently used to evaluate the clustering tendency of the nodes of a graph is the clustering coefficient measure [30]. The clustering coefficient of a node measures how much its neighbors are close to a clique, i.e., a complete subgraph. This measure assesses the number of triangles around a node divided by its number of possible cliques.

In this paper, we propose a new validation measure based on the clustering coefficient. As a result of the optimization of this measure, we introduce an algorithm in order to find graph clustering partitions with the maximum proposed clustering coefficient. This algorithm does not require the previous definition of the number of clusters in the partition. Computational experiments were carried out and the results obtained indicate a very good potential for finding good partitions using the proposed algorithm.

The rest of this paper is organized as follows. Section 2 presents the proposed measure based on clustering coefficient. In Sect. 3, a new clustering algorithm based on the optimization of the proposed measure is presented. Section 4 describes another measure, modularity, used in a comparison with the proposed measure. Finally, in Sect. 5 we evaluate the new measure and the quality of the partitions found by the introduced clustering algorithm. For such, we perform some experiments with several datasets. To sum up, Sect. 6 presents the main conclusions derived from the analysis of the experimental results.

2 Assessment measure

Let $G = (V, E)$ be a graph where V and E are, respectively, its set of nodes and edges. The number of nodes of G is n. Each edge is represented by a pair (i, j), where i and j are nodes from V. In this paper, the nodes are represented by natural numbers from 1 to n.

Consider $A = [a_{ij}]_{n\times n}$ to be the adjacency matrix of graph G. Each element of the adjacency matrix has a binary value, representing the relationship between two nodes. Thus, $a_{ij} = 1$ if nodes i and j are adjacent, i.e., if there is an edge linking node i to node j, and $a_{ij} = 0$ otherwise.

This paper deals with weighted graphs. Let $W = [w_{ij}]_{n\times n}$ be the weight matrix for the edges of a weighted graph G. The element w_{ij} of this matrix W is defined as the weight of the edge that links node i to node j. If there is no edge between a pair of nodes i and j, then $w_{ij} = 0$.

The degree of a node i, deg_i, from an unweighted or weighted graph, is calculated considering the number of its adjacent objects. It is given by (1).

$$deg_i = \sum_{j=1}^{n} a_{ij}. \tag{1}$$

A measure that evaluates the clustering tendency in graphs is known as *clustering coefficient* [30]. It is based on the analysis of three node cycles around a node i. A formulation of this measure for unweighted graphs is given by (2).

$$C_i = \frac{2\sum_{j=1}^{n-1}\sum_{k=j+1}^{n} a_{ij}a_{jk}a_{ik}}{deg_i(deg_i - 1)} \tag{2}$$

Note that $\sum_{\substack{j=1\\ j\neq i}}^{n-1}\sum_{\substack{k=j+1\\ k\neq i}}^{n} a_{ij}a_{jk}a_{ik}$ corresponds to the number of triangles around node i. The degree deg_i indicates the total number of neighbors of node i. The denominator measures the maximum possible number of edges that could exist between the vertices within the neighborhood. This measure evaluates the tendency of the nearest neighbors of node i to be connected to each other [20], as illustrated by Fig. 1.

This figure presents a subgraph (the first left-handed picture with no dashed lines) with a central black-colored node i and its four adjacent nodes (gray-colored). This means that the degree of i is 4 (the number of its incident edges highlighted by solid black lines). The maximum number of edges between the neighbors of i is 6, which is $4 \times 3/2$, as can be noticed in this picture. The number of triangles around the node i is also 6 (as shown in the 6 subgraphs with dashed lines), meaning that the clustering coefficient of this particular node i is 1.

According to [20], to extend this measure to weighted graphs, it is necessary to replace the part of the numerator, that indicates the number of triangles around node i, by the sum of the triangles intensities around node i. The formulation presented by the authors is:

$$C_i(W) = \frac{2\sum_{j=1}^{n-1}\sum_{k=j+1}^{n}(\tilde{w}_{ij}\tilde{w}_{ik}\tilde{w}_{jk})^{1/3}}{deg_i(deg_i - 1)}, \tag{3}$$

where $\tilde{w}_{ij} = w_{ij}/max\{w_{ij}|i, j \in Z, 1 \leq i, j \leq n\}$. It can be observed that $C_i \in [0, 1]$ and that $(\tilde{w}_{ij}\tilde{w}_{ik}\tilde{w}_{jk})^{1/3}$ indicates the cubic root of the weight product of the edge neighbors around node i. If $C_i = 1$, then the neighborhood of node i includes all combinations of triangles that can be formed and that each edge making part of this triangle has maximum weight. This index has a higher value when two nodes j and k that are adjacent to i are also adjacent to each other. If it is necessary to calculate the clustering coefficient of a graph, usually the average of the clustering coefficient of all nodes is considered.

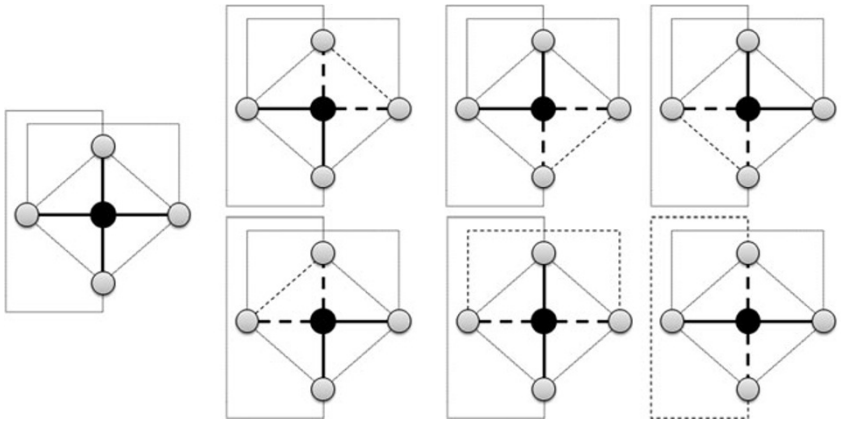

Fig. 1 Illustration of the clustering coefficient measure

Based on this weighted clustering coefficient, we propose a validation measure able to analyze the cluster tendency of a partition. For such, consider the following equation:

$$d_i = \sum_{j=1}^{n} a_{ij} x_{ij}, \tag{4}$$

where x_{ij} is a binary variable that assumes value 1, if the nodes i and j belong to the same cluster, and 0, otherwise. It must be observed that d_i is the number of nodes adjacent to i and that belong to the cluster with node i. The proposed clustering coefficient of a node i from a partition π of a weighted graph is given by the following equation:

$$Cc_i(\pi) = \frac{2\sum_{j=1}^{n-1}\sum_{k=j+1}^{n}(\tilde{w}_{ij}\tilde{w}_{jk}\tilde{w}_{ik})^{1/3}}{d_i(d_i-1)} y_{ijk}, \tag{5}$$

where y_{ijk} is a binary variable that assumes value 1 if the nodes i, j and k belong to the same cluster and 0, otherwise.

In this paper, we propose a solution method (heuristic) based on the optimization of the introduced clustering coefficient index. No proof with respect to the complexity of the problem whose goal is to detect the partition with the maximum proposed index is provided in this paper. In spite of this, according to our knowledge, no polynomial algorithm is always able to find such partition. Moreover, it is well known that the complexity of a variety of graph partitioning problems is NP-hard [6].

3 Solution method

Multilevel clustering algorithms are a special class of graph clustering algorithms that have been extensively used to find high quality clustering solutions. Some of these algorithms can be found in [4, 12, 13]. They are known for having three different phases: *coarsening*, *partitioning,* and *refinement*. The first phase, coarsening phase, compresses a graph by contracting its nodes. The second phase, partitioning phase, partitions the graph found in the coarsening phase. The last phase, refinement phase, expands the partitioned graph until it reaches its original structure by the simultaneous improvement of the current solution by the application of some method, like a local search.

In this paper, we propose a simple multilevel clustering algorithm. Its phases work as follows:

1. *Coarsening phase*: A sequence of matching's among the graph edges is performed until the graph reaches a predefined size.
2. *Partitioning phase*: In this phase, the coarsened graph is partitioned by the multi-level algorithm METIS [12, 13]. Many other partitioning algorithms, such as spectral clustering algorithms, were tested. However, METIS produced better results in a shorter time.
3. *Refinement phase*: The partition found in the previous phase is improved by a local search after each step of the expansion of the graph nodes by reverting the coarsening. This process continues until the original structure of the graph is obtained.

Details of each phase of the algorithm are presented next.

3.1 Coarsening phase

The coarsening phase consists in performing matching's of the edges from a graph. A matching on a graph G is defined by a set S of edges where, for every $(i, j) \in S$, there is no edge (k, j), with $k \neq i$ or (k, i) with $k \neq j$ that belongs to S. The maximum matching problem looks for a matching whose set of edges has the maximum total weight.

In order to perform the matchings in the proposed multilevel algorithm, a *heavy edge coarsening* was applied to the graph. A heavy edge coarsening randomly selects a node i nonincident to none edge from S. The heaviest edge incident to i and to a node j not incident to any edge from S is chosen to be matched, i.e., to belong to S. At each matching, the edge weights between the incident nodes from the matched edge and their adjacent nodes have to be updated.

For such, the edge weights between node i (and node j) and all its adjacent nodes have now to take into account the edge weights between these adjacent nodes with the other node in the matching, node j (and node i). An usual way to calculate the edge weight between a node i and one of its adjacent nodes, k, is summing up the edge weights connecting i with k and j with k.

In this paper, the coarsening phase is performed until the number of nodes becomes lower than or equal to the maximum value between 100 and $0.2n$. If a graph already has a number of nodes lower than or equal to 100, one iteration of the matching is performed. This final coarsened graph, named base graph, is partitioned in the next phase of the multilevel algorithm. The partitioning of this base graph is detailed in the next section.

3.2 Partitioning phase

According to multilevel algorithms, after the graph is coarsened, it is partitioned by a partitioning algorithm. In this paper, we applied a multilevel partitioning algorithm named METIS [12]. Other algorithms were tested, but METIS presented better final partitions.

As METIS requires the number of clusters to be defined a priori, we find partitions using METIS for $\{2, \ldots, 20\}$ clusters.

As a result, we have 19 different initial partitions to be refined. After the refinement phase, the partition with the best proposed clustering coefficient is returned as the final partition of the algorithm.

3.3 Refinement phase

In this phase, a local search procedure is applied to every partition found in the partitioning phase. Many different local search procedures were evaluated. We chose the local search with the best combination of efficiency and performance. For such, consider

$$t_i = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \frac{(\tilde{w}_{ij}\tilde{w}_{ik}\tilde{w}_{jk})^{1/3}}{d_i(d_i-1)} y_{ijk}.$$

Local Search (Solution)

Step 0: Read the graph and its partition to be refined by the local search. Make $it \leftarrow 0$.
Step 1: Unmark all nodes.
Step 2: Choose an unmarked node i whose t_i is lower than t_i'. t_i' is calculated in the same way as t_i, changing the solution variables as if node i belonged to a cluster c, different from where it belongs to. Mark node i. If $n < 500$, go to Step 3, else, go to Step 4.
Step 3: If moving node i to cluster c improve the current solution, go to Step 4. Else, go to Step 5.
Step 4: Move node i to cluster c.
Step 5: If all nodes are marked, make $it \leftarrow it + 1$ and go to Step 6, else, go to Step 2.
Step 6: If there was no improvement in this iteration or $it >$ 100, then go to Step 7. Else, go to Step 1.
Step 7: Return the most updated solution.

If the graph has more than 500 nodes, we eliminate Step 3, which is very expensive. However, when the graph has less than 500 nodes, it is viable to perform Step 3. This local search allows the number of clusters to change from the initial partition, when this produces a better solution.

As the order of the nodes to perform the matchings is random, different coarsened graphs can be achieved by the coarsening phase. For this reason, we perform all steps of the proposed multilevel algorithm 10 times. Only the best solution among these 10 runs is kept.

In summary, the proposed multilevel clustering algorithm (MLA-CC) has the following steps.

Multilevel algorithm (G)

Step 0: Read G. Make $it \leftarrow 0$.
Step 1: (Coarsening phase) Contract graph G as it was presented in the coarsening phase section.
Step 2: (Partitioning the base graph) Find the 19 partitions by METIS, as it was explained in Sect. 3.2.
Step 3: (Refinement phase) Apply the local search presented in Sect. 3.3 at each of the 19 partitions found in the previous step. Keep the best solution found.
Step 4: (Update solution) If $it = 0$ or the solution found in the previous step is higher than the overall best solution, then the overall best solution is updated by the best solution found in the previous step. Make $it \leftarrow it + 1$.
Step 5: If $it > 10$ then go to Step 6, else, go to Step 1.
Step 6: Return the best solution.

Next, a validation graph clustering measure extensively used in the last years will be presented. Partitions found by algorithms based on the optimization of this measure will be used to compare with the results of our algorithm.

4 Modularity index

The modularity index was proposed in [19]. It measures the clustering tendency of a graph partition, considering its probability in the same partition in a random graph with the same node degree sequence. To see how it works, let π be a partition from a graph G. Consider the following equation:

$$q(\pi) = \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij} x_{ij}.$$

If G is a weighted graph, $2m = \sum_{l=1}^{n} \sum_{k=1}^{n} w_{lk}$, and r_{ij} defined by the following equation:

$$r_{ij} = w_{ij} - \frac{\sum_{k=1}^{n} w_{ik} \sum_{k=1}^{n} w_{jk}}{\sum_{l=1}^{n} \sum_{k=1}^{n} w_{lk}}. \quad (7)$$

Heuristics based on the search of the partition with maximum modularity have been extensively proposed in literature [3]. Due to the quality of the partitions found by these algorithms, they are largely used as the main references for graph clustering.

5 Computational experiments

In order to analyze the performance of the proposed algorithm, we performed two sets of experiments: the first with artificial graphs and the second with real datasets converted into similarity graphs. Each one of the experiments is explained in the next sections. In these experiments, we compare our heuristic with the following graph clustering algorithms from the literature: a spin glass-based algorithm (Spinglass) [23], a fast greedy modularity-based algorithm (FastGreedy) [3] and a walktrap algorithm (Walktrap) [21]. They were all proposed for graph clustering problems and do not require the definition of the number of clusters. We used their implementation from the R-project [29] in the package *igraph*. All of them use the validation measure adopted by the authors, the modularity measure (the partition with the best modularity in the iterations of the algorithms is always chosen). The comparison of the algorithms is based on the real classification of the datasets. For such, we use the *Adjusted Rand Index* (ARI), proposed in [10]. This index allows to compare the similarity between the partition found by an algorithm for a given dataset with the real classes. The value for this index is better when its value is higher. The minimum and maximum values achieved by this measure are, respectively, -1 and 1.

5.1 First experiment

In order to analyze the behavior of the proposed solution method, MLA-CC, experiments were carried out with sixty artificial modular graphs, generated using the function *SimDataAffiliation* from the package *Statistical Inference for Modular Networks* (SIMoNe). The parameters used to generate the graphs were set in order to make them highly modular. We generated six graphs for each of the following number of nodes: 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000. For each number of nodes, there is one graph with 2, 3, 4, 5, 10, and 20 modules (clusters in the structure of the graph).

As a result of this experiment, we observed that the proposed algorithm presented ARI values worse than the other algorithms used in the comparison. The analysis of these results showed that, in general, a poor performance of MLA-CC was observed when the input graph was not complete. Additional experiments were then carried out using complete graphs. In these new experiments, our algorithm performed better than the other algorithms used in the comparison. These results suggest that a transformation of a non complete graph into a complete graph could improve the performance of the proposed algorithm in the other datasets. However, this transformation may not be efficient, since to work with a sparse matrix is, in general, computationally easier than with non sparse matrices. We tested a conversion of non complete graphs to a complete graph version using an approach based on the pairwise shortest paths between the nodes. Even though the proposed algorithm achieved better results with these converted graphs than with the original noncomplete graphs, the performance of the resulting algorithm was inferior to the performance observed in the other evaluated algorithms. Table 1 shows these results (using graphs transformed to complete graphs by a shortest path approach for MLA-CC and raw graphs for the other algorithms).

Table 1 Average ARI values of the partitions found by the algorithms regarding the clustering labels provided by the data generator. Each average corresponds to the mean ARI of the partitions for all artificial graphs with the same number of nodes

Dataset (♯nodes)	MLA-CC	Spinglass	Walktrap	FastGreedy
100	0.68	0.59	0.66	0.89
200	0.30	0.41	0.50	0.95
300	0.02	0.21	0.33	0.95
400	0.11	0.59	0.83	0.51
500	0.16	0.91	0.99	0.86
600	0.16	0.99	0.99	0.95
700	0.09	0.80	1.00	0.78
800	0.09	0.83	0.83	0.87
900	0.15	0.88	1.00	0.80
1000	0.13	0.84	1.00	0.79

The results from Table 1 show that the proposed algorithm has ARI values similar to the other algorithms only for the smallest number of nodes. Its performance becomes significantly worse than the other algorithms with the increase of the number of nodes. These results suggest that the proposed algorithm should not be used in noncomplete graphs. However, additional experiments using real datasets indicated a niche where the proposed algorithm has a very good potential. These experiments are presented in the next section.

5.2 Second experiment

In this second experiment, fourteen real datasets were used. Eleven of them are biological datasets and three are benchmark datasets largely used in cluster analysis. The main aspects of these datasets are explained in Tables 2 and 3.

In order to construct a similarity graph from the datasets observed in Tables 2 and 3, we used the following strategy. Let G be a complete graph from a given dataset. Each object from the dataset represents a node from G, whereas the

Table 2 Table of the description of the biological datasets used in the second experiment

Golub [8] is a dataset with 3,571 gene expression levels of 47 tissues of acute lymphocyte leukemia (ALL) samples and 25 tissue samples of acute myeloid leukemia (AML). ALL may be classified in 2 classes: B-lineage with 38 samples and T-lineage with 9 samples
BreastA [28] and BreastB [31] are datasets with, respectively, 98 and 48 samples, of breast tumor generated using one-channel oligonucleotide and two-channel microarrays. The version employed is pre-processed with 1,213 attributes each. Regarding the classification, in [9] can be observed an analysis of BreastA dividing it in 3 classes with 11, 51, and 36 samples. BreastB, like in [18], was decomposed into 2 classes regarding the estrogen receptor (ER). In this dataset, there are 25 samples of positive ER (ER+) 24 samples of negative (ER−). The collection of tumors consists of 13 samples of ER+ lymph node (LN)+ tumors, 12 samples of ER− LN+ tumors, 12 samples of ER+ LN− tumors, and 12 samples of ER− LN− tumors
DLBCLC [26] is a dataset with 58 samples of patients with Diffuse large B cell lymphoma (DLBCL). Its number of attributes is 3,795, which corresponds to a preprocessed version from [9]. The classes of this dataset correspond to the cured DLBCLs (32 samples) and the fatal/refractory tumors (26 samples)
Leukemia [32]: This dataset has 327 samples with 271 gene expression levels. It can be classified into 7 classes: BCR-ABL (15 samples), E2A-PBX1 (27 samples), Hiperdiploid>50 (64 samples), MLL (20 samples), T-ALL (43 samples), TEL-AML1 (79 samples) and others (79 samples). An alternative grouping divides the example in 3 more general subgroups, in which the first includes B-ALL, which consists of samples BCR, E2A, TEL, and MLL; the second is composed of T-ALL; and last one consists of "others" type
Lung [1]: This dataset consists of 197 lung tumor samples with 1,000 gene expression levels. Its classification into 4 types of cancer includes 139 samples of adenocarcinomas, 21 samples of carcinomas from squamous cells and 20 samples of carcinoids. A pre-processed version from [16] was used
MultiA [27] is a preprocessed cancer tissue dataset [9]. It has the same examples and classes as Novartis, except for the number of attributes, which is higher, 5565
MultiB [22] is, as well as MultiA, a cancer tissue dataset. It is also a pre-processed version found in [9] and has 32 samples with 5,565 attributes. This dataset has 4 classes corresponding to different types of tissues: 5 breast tissues, 9 prostate tissues, 7 lung tissues and 11 colon tissues
Novartis [16, 27]: This dataset has 1,000 gene expression levels from 103 cancer tissue samples. The classification comes from their origin: 26 from breast, 26 from prostate, 28 from lung and 23 from colon
MiRNA [14]: This dataset corresponds to 218 mammal tissue samples of human and tumor origins with gene expression profiles of 217 microRNAs. The samples are classified into 20 classes with 6, 15, 10, 11, 3, 9, 18, 7, 19, 10, 8, 5, 14, 2, 26, 28, 8, 8, 3 and 8 samples according to their origin
Yeast [17] is a 8 attribute dataset with 1484 yeast proteins samples. As it can be observed in [17], the dataset can be classified into 10 classes regarding the localization site of proteins, with 463, 429, 244, 163, 51, 44, 37, 30, 20, and 5 objects

Table 3 This table shows the description of the non-biological datasets used in the experiments

Glass [5] is a classical 9 attribute dataset of criminal investigation scenes with 214 samples. The samples may be divided into 6 classes according to the physic chemical properties, with 70, 76, 17, 13, 9 and 29 examples
Iris [7] is a 4 classical attribute dataset with 150 examples of 3 species of the Iris flower: 150 samples of Iris setosa, 150 samples of Iris virginica and 150 samples of Iris versicolor
Simulated6 [16] is an artificial dataset with 60 samples with 600 genes as attributes. This dataset can be divided into 6 classes with 8, 12, 10, 15, 5 and 10 samples. Each class is defined by 50 distinct genes, which are uniquely regulated for each class

edges linking pairs of nodes are weighted according to the weight matrix W. The weights are calculated according to the Euclidean distance, using (8).

$$w_{ij} = 1 - \frac{d_{ij}}{d_{max}}, \tag{8}$$

where d_{ij} is the Euclidean distance between objects i and j, and d_{max} is the maximum distance between any pair of objects from the dataset. Other ways for constructing a graph from a dataset could be used. However, many studies, like [15], claim that the discovery of the most suitable way to represent a dataset in a graph is not an easy task. Therefore, for simplicity, in this paper, we just used this strategy for constructing graphs from datasets.

5.2.1 Results of the second experiment

Table 4 shows the for the four investigated algorithms. The columns Dataset and #C indicate, respectively, the tested dataset and its number of classes according to its available real classifications. The column #CL shows the number of

Table 4 ARI values of the partitions found by the algorithms. The best ARI for each dataset is highlighted in bold

Dataset	#C	MLA-CC		Spinglass		Walktrap		FastGreedy	
		#CL	ARI	#CL	ARI	#CL	ARI	#CL	ARI
Golub	2	3	0.425	3	0.441	1	0.000	2	**0.837**
Golub	2	3	0.011	3	0.011	1	0.000	2	**0.032**
Golub	3	3	0.395	3	0.428	1	0.000	2	**0.645**
Golub	4	3	0.318	3	0.341	1	0.000	2	**0.630**
BreastA	3	2	0.654	3	0.712	3	0.562	2	**0.741**
BreastB	4	3	0.202	2	**0.269**	1	0.000	2	**0.269**
DLBCLC	2	2	−0.017	2	−0.018	1	0.000	2	−0.018
Glass	6	6	0.189	3	0.170	3	0.238	2	**0.261**
Iris	3	4	**0.638**	3	0.513	2	0.568	2	0.523
Leukemia	3	9	**0.236**	5	0.138	4	0.200	3	0.041
Leukemia	7	9	**0.689**	5	0.521	4	0.550	3	0.392
Lung	4	6	**0.303**	3	0.151	3	0.254	2	0.117
MultiA	4	5	**0.702**	3	0.664	3	0.580	3	0.633
MultiB	4	2	**−0.027**	2	−0.027	1	0.000	2	−0.027
Novartis	4	4	**0.946**	4	**0.946**	2	0.331	3	0.612
Simulated6	6	5	**0.871**	3	0.326	2	0.180	3	0.326
MiRNA	20	8	**0.315**	2	0.097	2	0.094	2	0.098
Yeast	10	3	**0.156**	8	0.084	2	0.035	2	0.082

clusters of the partitions found by the indicated algorithm. These results are concerned with the real classification of the datasets provided in literature.

As can be seen on Table 4, according to the ARI measure, MLA-CC found the best ARI more times than the other algorithms. In the cases where one of the other algorithms obtained better results than MLA-CC, their ARI values were very close (except for the dataset Golub). Therefore, considering these datasets, the proposed graph clustering algorithm for weighted graphs produced good results. Moreover, it can be noticed that the majority of the results, for all algorithms, were lower than 0.4. This fact suggests that in many cases the dataset had not a clear clustering structure, or that it is not easily identified. Taking as example the dataset MiRNA, one may observe that the partitions found by the other algorithms have a very low resemblance with the real classification of the dataset (since their ARI values were around 0.098). The partition found by MLA-CC for this same dataset has a much better suitability to the real data classification (in this case, the ARI was 0.315, more than three times higher than the other algorithms). Another dataset for which the proposed algorithm achieved much better results, when compared with the other algorithms, was the dataset Simulated6. On the other hand, for the dataset Golub, the FastGreedy algorithm found a partition with a much better average ARI value than the other algorithms.

Table 5 Table of objective function values

Dataset	CC
Golub	0.590159
BreastA	0.78431
BreastB	0.575682
DLBCLC	0.630128
Glass	0.793655
Iris	0.839137
Leukemia	0.642047
Lung	0.687603
MultiA	0.490546
MultiB	0.598776
Novartis	0.543799
Simulated6	0.685092
MiRNA	0.715988
Yeast	0.776264

Table 5 presents the resulting clustering coefficient proposed in this paper for each partition found by MLA-CC. Observe that, in most cases, the clustering coefficient of the partitions is higher than 0.5. This means that the partitions found have a good cluster tendency, and that the algorithm proposed to obtain these partitions had a good performance. In particular, the partition found for the dataset Iris had high clustering coefficient. This means that regarding this validation measure the partitions obtained by the algorithm had a very good quality, since the possible maximum value is 1.

Regarding the computational efficiency of the proposed heuristic, there is a significant impact of both the number of natural clusters and the number of nodes in the dataset on the computational time. In general, the lower the number of natural clusters, the higher the computational time. Although there is no clear pattern associating the number of clusters and the computational time, we estimate an increase of 10% in running time for each cluster (e.g., a dataset with 3 clusters and 100 nodes took 4 seconds, whereas the running time for a dataset with the same number of nodes and 2 clusters was about 4.5 seconds). There is also a large impact of the number of nodes on the computational time necessary to find the solution (although we can make no assertion about its relation to the number of nodes, it is approximately twice the solution time for a dataset with one hundred less nodes). In comparison with the other algorithms, MLA-CC took approximately the same time as Spinglass to find the partitions (both took the largest computational times among all tested algorithms, about ten times slower than FastGreedy and Walktrap).

6 Final remarks

This paper presented a novel graph clustering algorithm based on a well-studied measure known as clustering coefficient. The measure consists in the analysis of the connectivity of a node regarding the connectivity of its neighbors and is commonly used to measure the clustering tendency of the nodes in a graph. In this paper, a variation of an existing weighted version for clustering coefficient is proposed in order to evaluate the quality of the partitions of weighted graphs. Using this variation, an algorithm based on the optimization of the proposed measure is presented.

In order to validate the quality of the proposed algorithm, we compared its results with the results of other graph clustering algorithms from literature. The comparison was based on a measure that evaluates the fitness of a pair of partitions (the partitions found by algorithms and the real classification of the data). In a first experiment with artificial modular graphs, we observed a poor performance of the proposed algorithm, with much worse results than from the literature for noncomplete graphs. By studying the behavior of the results of our algorithm, we discovered that it worked better when the case study was a complete graph. Then, in a second experiment with complete graphs, we attested that the partitions found by the proposed algorithm had a better quality if compared with the partitions obtained with algorithms from literature considering the classification of the real datasets. Therefore, the proposed algorithm for weighted graphs is a valuable novel contribution for graph clustering for complete graphs.

Acknowledgements The authors gratefully acknowledge FAPESP, CAPES, and CNPq for their financial support.

Table 6 Weight matrix of the toy example, where the matching of the coarsening phase is represented by the edges highlighted by gray boxes

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0.8	0.9	1.0	0.9	0.9	0.9	0.6	0.2	0.3	0.2	0.2	0.3	0.2	0.2	0.2	0.3	0.2	0.2	0.2
2	–	0	0.8	0.8	0.7	0.9	0.9	0.4	0.2	0.2	0.2	0.2	0.2	0.1	0.1	0.1	0.2	0.2	0.1	0.2
3	–	–	0	0.9	0.8	0.8	0.8	0.4	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.2	0.1	0.1	0.1
4	–	–	–	0	0.9	0.9	0.9	0.4	0.1	0.2	0.1	0.2	0.2	0.2	0.1	0.1	0.2	0.1	0.1	0.2
5	–	–	–	–	0	0.9	0.9	0.5	0.2	0.2	0.1	0.1	0.2	0.2	0.1	0.1	0.2	0.1	0.2	0.2
6	–	–	–	–	–	0	0.9	0.4	0.2	0.2	0.1	0.2	0.2	0.2	0.1	0.2	0.2	0.2	0.1	0.1
7	–	–	–	–	–	–	0	0.4	0.2	0.2	0.1	0.2	0.2	0.2	0.1	0.2	0.2	0.2	0.1	0.2
8	–	–	–	–	–	–	–	0	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.5	0.6	0.6	0.5	0.5
9	–	–	–	–	–	–	–	–	0	0.9	0.8	0.9	0.9	0.9	0.8	0.9	0.9	0.9	0.9	0.8
10	–	–	–	–	–	–	–	–	–	0	0.9	1.0	0.9	1.0	0.9	0.9	0.9	0.9	0.8	1.0
11	–	–	–	–	–	–	–	–	–	–	0	0.9	0.9	0.8	0.8	0.8	0.8	0.8	0.8	0.9
12	–	–	–	–	–	–	–	–	–	–	–	0	1.0	0.9	0.9	0.9	1.0	0.9	0.9	0.9
13	–	–	–	–	–	–	–	–	–	–	–	–	0	0.9	0.8	1.0	0.9	1.0	0.9	0.9
14	–	–	–	–	–	–	–	–	–	–	–	–	–	0	0.8	0.9	0.9	0.9	0.8	1.0
15	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	0.8	0.9	0.8	0.8	0.9
16	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	0.9	0.8	0.8	0.9
17	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	0.9	0.9	1.0
18	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	0.8	0.9
19	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0	0.9
20	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0

Table 7 The pairwise weight matrix of the coarsened graph from the example

	[12,13]	[20,10]	[2,7]	[8,17]	[11,19]	[14,18]	[4,1]	[15,9]	[16,6]	[3,5]
[12,13]	0	1.11	0.13	0.89	1.08	1.12	0.14	1.07	0.65	0.10
[20,10]	–	0	0.15	0.86	1.04	1.13	0.16	1.03	0.61	0.09
[2,7]	–	–	0	0.27	0.08	0.12	1.04	0.08	0.56	0.95
[8,17]	–	–	–	0	0.77	0.83	0.34	0.79	0.53	0.28
[11,19]	–	–	–	–	0	0.99	0.06	0.99	0.49	0.06
[14,18]	–	–	–	–	–	0	0.14	1.03	0.57	0.03
[4,1]	–	–	–	–	–	–	0	0.08	0.58	1.09
[15,9]	–	–	–	–	–	–	–	0	0.55	0.05
[16,6]	–	–	–	–	–	–	–	–	0	0.51
[3,5]	–	–	–	–	–	–	–	–	–	0

Appendix: a running example

This Appendix presents one iteration of a toy example of MLA-CC with a small dataset. In this example, we used the first 20 objects of the dataset Simulated6. Each step of the algorithm is presented in the next sections.

7.1 Coarsening phase

As we are dealing with a complete graph example, let us represent its weighted edges on a weight matrix. The resulting matching between the nodes of this example is represented in the weight matrix illustrated in Table 6 by the gray boxes.

According to Table 6, the edge between nodes 4 and 1 is matched, since the box corresponding to the first row and forth column is marked by the gray color. In Table 7, the node of the coarsened graph corresponding to the matching of this edge is referred as [4,1]. Thus, one has a graph with 10 nodes (resulted from the coarsening of the original graph by the matching) whose weighted edges are showed in Table 7 (calculated as explained before).

7.2 Partitioning phase

The coarsened graph is then partitioned at this phase of the algorithm. In this paper, we used a strategy that generates partitions from 2 to 10 clusters using METIS (for a larger dataset, it would be possible to generate 2 to 19 clusters, as presented in the algorithm description). The resulting partitions are presented in the matrix in Table 8, where the ith row represents the node i from the coarsened graph, and each number of the matrix is the class label of the resulting partition (represented in each column). Using these 9 partitions, the refinement phase of the proposed algorithm is performed.

7.3 Refinement phase

In this phase of the algorithm, at each uncoarsening step (in this example, it is just one step, due to the small size of the dataset), a local search is applied to the current partition. For each of the 9 partitions, the uncoarsening of the coarsened graph followed by a local search is performed (updating the labels). For example, in the forth partition of the coarsened graph, the first row corresponds to the first node of the coarsened graph that corresponds to the nodes 12 and 13 of the original graph (indicated as [12,13]). Both of them will receive the label 1, as indicated in Table 9.

At this phase, the local search is applied to the partition of the current iteration. As already described, the local search consists in the transfer of nodes between the clusters in the partition of a graph in such a way that the transfer produces a partition with a better clustering coefficient-based index. The refinement phase of the forth partition (the one that produces the solution with best proposed index in the end of the procedure) is summarized in Table 10.

It is important to remind that this example corresponds solely to one of a total of 50 iterations. At each iteration, a different matching is used (generated with a different seed).

Table 8 Table with the partitions found by METIS for the coarsened graph

	Partitions								
Nodes	♯ 1	♯ 2	♯ 3	♯ 4	♯ 5	♯ 6	♯ 7	♯ 8	♯ 9
[12,13]	1	3	1	4	2	4	1	9	4
[20,10]	1	3	1	4	2	4	1	7	5
[2,7]	2	1	3	1	5	2	5	1	9
[8,17]	1	2	2	3	1	6	3	5	1
[11,19]	1	3	2	4	1	6	3	5	1
[14,18]	1	3	1	3	2	4	1	9	5
[4,1]	2	1	3	1	6	3	6	2	9
[15,9]	1	2	1	5	3	5	2	8	3
[16,6]	2	2	4	2	4	1	8	4	8
[3,5]	2	1	3	1	6	3	6	2	7

Table 9 This table shows the uncoarsened partition number four

Nodes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Label	1	1	1	1	1	2	1	3	5	4	4	4	4	3	5	2	3	3	4	4

Table 10 The partitions found at each of the 14 iterations of the local search from the refinement phase of MLA-CC. The movements of the local search are highlighted by gray boxes

	Iterations													
Nodes	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	2	5	5	5	5	5	5	5	5	5	5	5	5	5
2	1	1	2	2	2	2	2	2	2	3	4	4	4	4
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	3	3	3	3	3	3	3	3	3	3	3
5	1	1	1	1	1	1	1	1	1	1	1	2	2	2
6	2	2	2	2	2	2	2	2	2	2	2	2	2	2
7	1	1	1	1	1	1	1	1	1	1	1	1	1	2
8	3	3	3	3	2	2	2	2	2	2	2	2	2	2
9	5	5	5	5	5	5	5	5	5	5	5	5	5	5
10	4	4	4	4	4	1	1	1	1	1	1	1	1	1
11	4	4	4	4	4	4	1	1	1	1	1	1	1	1
12	4	4	4	4	4	4	4	4	4	4	4	4	1	1
13	4	4	4	4	4	4	4	4	4	4	4	4	4	4
14	3	3	3	3	3	3	3	3	3	3	3	3	3	3
15	5	5	5	5	5	5	5	5	5	5	5	5	5	5
16	2	2	2	2	2	2	2	2	2	2	2	2	2	2
17	3	3	3	3	3	3	3	1	2	2	2	2	2	2
18	3	3	3	3	3	3	3	3	3	3	3	3	3	3
19	4	4	4	4	4	4	4	4	4	4	4	4	4	4
20	4	4	4	4	4	4	4	4	4	4	4	4	4	4

References

1. Bhattacharjee A, Richards WG, Staunton J, Li C, Monti S, Vasa P, Ladd C, Beheshti J, Bueno R, Gillette M, Loda M, Weber G, Mark EJ, Lander ES, Wong W, Johnson BE, Golub TR, Sugarbaker DJ, Meyerson M (2001) Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. Proc Natl Acad Sci USA 98(24):13790–13795
2. Boginski V, Butenko S, Pardalos PM (2006) Mining market data: a network approach. Comput Oper Res 33:3171–3184
3. Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70(6):066111
4. Dhillon IS, Guan Y, Kulis B (2007) Weighted graph cuts without eigenvectors a multilevel approach. IEEE Trans Pattern Anal Mach Intell 29(11):1944–1957
5. Evett IW, Spiehler EJ (1987) Rule induction in forensic science. In: KBS in government, online publications, pp 107–118
6. Feder T, Hell P, Klein S, Motwani R (1999) Complexity of graph partition problems. In: 31ST ANNUAL ACM STOC. Plenum, New York, pp 464–472
7. Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugen 7:179–188
8. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439):531–537
9. Hoshida Y, Brunet JP, Tamayo P, Golub TR, Mesiro JP (2007) Subclass mapping: identifying common subtypes in independent disease data sets. PLoS ONE 2(11):e1195
10. Hubert L, Arabie P (1985) Comparing partitions. J Classif 2:193–218
11. Huttenhower C, Flamholz AI, Landis JN, Sahi S, Myers CL, Olszewski KL, Hibbs MA, Siemers NO, Troyanskaya OG, Coller HA (2007) Nearest neighbor networks: clustering expression data based on gene neighborhoods. BMC Bioinform 8:250
12. Karypis G, Kumar V (1996) Parallel multilevel graph partitioning. In: Proceedings of the international parallel processing symposium
13. Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20(1):359–392
14. Lu J, Getz G, Miska EA, Alvarez-Saavedra E, Lamb J, Peck D, Sweet-Cordero A, Ebert BL, Mak RH, Ferrando AA, Downing JR, Jacks T, Horvitz RR, Golub TR (2005) Microrna expression profiles classify human cancers. Nature 435(7043):834–838
15. Maier M, von Luxburg U, Hein M (2009) Influence of graph construction on graph-based clustering measures. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in neural infor-

mation processing systems, vol 21, pp 1025–1032. Curran, Red Hook
16. Monti S, Tamayo P, Mesirov J, Golub T (2003) Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. Kluwer Academic, Dordrecht. Tech rep, Broad Institute/MIT
17. Nakai K, Kanehisa M (1991) Expert system for predicting protein localization sites in gram-negative bacteria. Proteins 11:95–110
18. Nascimento MCV, Toledo FMB, Carvalho ACPLF (2010) Investigation of a new GRASP-based clustering algorithm applied to biological data. Comput Oper Res 37:1381–1388
19. Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69:026113
20. Onnela JP, Saramäki J, Kertész J, Kaski K (2005) Intensity and coherence of motifs in weighted complex networks. Phys Rev E 71:065(R), 103(R)
21. Pons P, Latapy M (2005) Computing communities in large networks using random walks. In: Computer and information sciences—ISCIS 2005, pp 284–293
22. Ramaswamy S, Tamayo P, Rifkin R, Mukherjee S, Yeang CH, Angelo M, Ladd C, Reich M, Latulippe E, Mesirov JP, Poggio T, Gerald W, Loda M, Lander ES, Golub TR (2001) Multiclass cancer diagnosis using tumor gene expression signatures. Proc Natl Acad Sci USA 98(26):15,149–15,154
23. Reichardt J, Bornholdt S (2006) Statistical mechanics of community detection. Phys Rev E 74:016 110
24. Schaeffer SE (2007) Graph clustering. Comput Sci Rev 1:27–64
25. Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22:888–905
26. Shipp MA, Ross KN, Tamayo P, Weng AP, Kutok JL, Aguiar RCT, Gaasenbeek M, Angelo M, Reich M, Pinkus GS, Ray TS, Koval MA, Last KW, Norton A, Lister TA, Mesirov J (2002) Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. Nat Med 8:68–74
27. Su AI, Cooke MP, Ching KA, Hakak Y, Walker JR, Wiltshire T, Orth AP, Vega RG, Sapinoso LM, Moqrich A, Patapoutian A, Hampton GM, Schultz PG, Hogenesch JB (2002) Large-scale analysis of the human and mouse transcriptomes. Proc Natl Acad Sci USA 99:4465–4470
28. van 't Veer LJ, Dai H, van de Vijver MJ, He YD, Hart AA, Mao M, Peterse HL, van der Kooy K, Marton MJ, Witteveen AT, Schreiber GJ, Kerkhoven RM, Roberts C, Linsley PS, Bernards R, Friend SH (2002) Gene expression profiling predicts clinical outcome of breast cancer. Nature 415(6871):530–536
29. Venables WN, Smith DM (2010) An introduction to R. R Development Core Team, The R Foundation for Statistical Computing, version 2.11.1
30. Watts D, Strogatz S (1998) Collective dynamics of small-world networks. Nature 393:440
31. West M, Blanchette C, Dressman H, Huang E, Ishida S, Spang R, Zuzan H, Olson JA, Marks JR, Nevins JR (2001) Predicting the clinical status of human breast cancer by using gene expression profiles. Proc Natl Acad Sci USA 98(20):11462–11467
32. Yeoh EJ, Ross ME, Shurtleff SA, Williams WK, Patel D, Mahfouz R, Behm F, Raimondi SC, Relling MV, Patel A, Cheng C, Campana D, Wilkins D, Zhou X, Li J, Liu H, Pui CH, Evans WE, Naeve C, Wong L, Downing J (2002) Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell 1:133–143