A hybrid evolutionary algorithm for finding low conductance of large graphs

Zhi Lu [a], Jin-Kao Hao [a,b,*], Qinghua Wu [c]

[a] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[b] *Institut Universitaire de France, 1 rue Descartes, 75231 Paris, France*

[c] *School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China*

Future Generation Computer Systems,
https://doi.org/10.1016/j.future.2019.12.049

Abstract

Given an undirected graph $G = (V, E)$ with vertex set V and edge set E, the minimum conductance graph partitioning problem is to partition V into two subsets $s = (A, \bar{A})$ such that its quotient cut is minimized. This problem arises in a number of significant real-world applications; however, it is known to be NP-hard and thus computationally challenging. We present in this work a hybrid evolutionary algorithm for finding low conductance of large graphs. To ensure a high search efficiency and efficacy, the algorithm relies on the general memetic search framework and integrates specialized search components (e.g., fast local optimization with a progressive neighborhood and quality-and-diversity based pool management). Computational experiments on a set of 60 large-scale real-world benchmark instances in the literature with up to 23 million vertices lead to the following main findings. First, the algorithm competes very favorably with the state-of-the-art methods and is able to find efficiently solutions of low conductance for the tested instances. Second, the dedicated search components are essential to ensure the high performance of the algorithm. Third, the algorithm can further ameliorate the solutions obtained by the well-known max-flow quotient-cut improvement algorithm. Finally, we illustrate an application of using the algorithm to detect meaningful community structures in complex networks.

Keywords: Conductance minimization; Cheeger number; Hybrid search; Metaheuristics; Large graph optimization.

[*] Corresponding author. Email address: jin-kao. hao@univ-angers.fr (J.K. Hao)

1 Introduction

Graph partitioning is a general and convenient tool that can be used to model a variety of practical problems. Let $G = (V, E)$ be an undirected and connected graph with vertex set V and edge set $E \subseteq V \times V$, graph partitioning generally involves finding a partition (or a cut) $(A, \bar{A})$ of V (i.e., $A \cup \bar{A} = V$ and $A \cap \bar{A} = \emptyset$) in accordance with an optimization objective. A typical example is the well-known graph 2-way partitioning problem that aims to minimize the cut edges (i.e., the edges with endpoints in different partition subsets) of the partition. In this work, we are interested in the following conductance (also called quotient cut) criterion:

$$\Phi(s) = \frac{|cut(s)|}{min\{vol(A), vol(\bar{A})\}} \tag{1}$$

where $s = (A, \bar{A})$ is a cut of G, $cut(s)$ is the set of cut edges of partition s (called cut-set), and $vol(.)$ is the volume of subset A or $\bar{A}$ given by the sum of degrees of the vertices of the subset. Fig. 1 [28] shows a graph G with vertex set $V = \{a, b, \ldots, g\}$ and a partition $s = (A, \bar{A})$ with $A = \{c, d, f, g\}$ and $\bar{A} = \{a, b, e\}$. Since $vol(A) = 15$, $vol(\bar{A}) = 7$ and $|cut(s)| = 5$, this partition has a conductance $\Phi(s) = 5/7 = 0.71$.

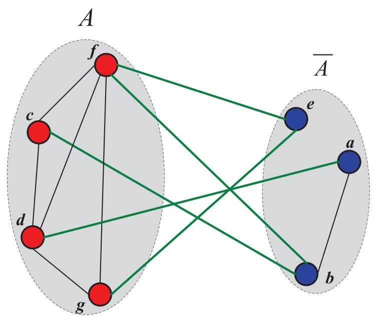

Fig. 1. For the given graph G, we show a partition $s = (A, \bar{A})$ where $A = \{c, d, f, g\}$ and $\bar{A} = \{a, b, e\}$ which has a conductance $\Phi(s)$ of 0.71 [28].

For a given graph G, its conductance is the lowest conductance over all possible partitions of the graph. The minimum conductance graph partitioning problem (MC-GPP) is then to determine the conductance of a general graph. In mathematics and statistical physics, the conductance of a graph is also called the Cheeger constant (Cheeger number or isoperimetric number) [9]. Intuitively, finding the conductance of a graph is equivalent to minimize the relative number of edges going out of the partition subset A with respect to its size in terms of volume (or the size of $\bar{A}$ if $\bar{A}$ has a smaller size).

Inspired by [18], we introduce the following mathematical model to formulate

MC-GPP.

Let $s = (A, \bar{A})$ be a partition of G, we define, for each vertex $i \in V$, a binary variable x_i such that

$$x_i = \begin{cases} 1 \text{ , if } i \in A \\ 0 \text{ , if } i \in \bar{A} \end{cases} \tag{2}$$

We define, for each edge $\{i, j\} \in E$, two additional binary variables: $z_{ij} = 1$ if exactly one of its endpoints i or j is in A; $y_{ij} = 1$ if both i and j are in A,

$$z_{ij} = \begin{cases} 1, \text{ if } i \in A, j \in \bar{A}, \text{ or } i \in \bar{A}, j \in A \\ 0, \text{ if } i, j \in A, \text{ or } i, j \in \bar{A} \end{cases} \tag{3}$$

$$y_{ij} = \begin{cases} 1, \text{ if } i, j \in A \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

Let C denote the number of the edges crossing the cut (i.e., $C = |cut(s)|$), let I denote the number of edges whose endpoints are in A,

$$C = \sum_{\{i,j\} \in E} z_{ij} \tag{5}$$

$$I = \sum_{\{i,j\} \in E} y_{ij} \tag{6}$$

Then, MC-GPP can be state as the following minimization problem:

$$\text{Minimize} \quad f(s) = \frac{C}{\min\{C + 2 * I, 2 * |E| - (C + 2 * I)\}} \tag{7}$$

$$\text{Subject to} \quad x_i - x_j \leq z_{ij} \text{ and } x_j - x_i \leq z_{ji}, \ \forall \{i, j\} \in E \tag{8}$$

$$y_{ij} \leq x_i \text{ and } y_{ij} \leq x_j, \ \forall \{i, j\} \in E \tag{9}$$

$$1 \leq \sum_{\{i,j\} \in E} y_{ij} \leq |E| - 1 \tag{10}$$

$$x_i, x_j \in \{0, 1\}, \ \forall i, j \in V \tag{11}$$

$$z_{ij}, y_{ij} \in \{0, 1\}, \ \forall \{i, j\} \in E \tag{12}$$

where $s = (A, \bar{A})$ is a cut. Eq. (7) states the minimization objective which is equivalent to Eq. (1). Constraint (8) ensures that the endpoints of each edge crossing the cut are located in different subsets of the cut, while Constraint (9) imposes that the endpoints of an edge $\{i, j\}$ which does not cross the cut belong to the same subset. Constraint (10) ensures that subset A contains at least one edge with its endpoints in A, and at least one edge crossing the cut (thus no subset is empty). Constraints (11) and (12) indicate that the corresponding variables take binary values.

MC-GPP has various significant real-world applications. For instance, in [44], conductance was used to identify ground-truth communities in large real-world networks where nodes organize into densely linked communities with edges appearing with high concentration among the members of the community. In [8], a graph partitioning approach with minimum conductance was used to find bottlenecks in complex networks where the existence of low conductance minima indicates bottlenecks. Other relevant applications of MC-GPP include clustering [10,35,39,46], community detection in complex networks [12,23,41], analysis of protein-protein interaction networks in biology [42], and image segmentation in computer vision [36].

Meanwhile, MC-GPP is a computationally challenging problem from the perspective of solution methods because it is known to be NP-hard in terms of complexity [37]. Our literature review (see Section 2) indicates that unlike other graph partition problems, few practical methods are available for solving MC-GPP. Indeed, most existing methods rely on the approximation framework. These methods are either specific to a particular application (e.g., clustering, community identification) or become unpractical for large and massive graphs due to their high computational complexity. One observes that solution methods based on modern metaheuristics remain scarce, even though they have shown to be powerful tools for solving difficult optimization problems. Indeed, heuristic algorithms were largely neglected and research on such methods is still in its infancy. This work thus aims to enrich the toolkit of practical solution methods for MC-GPP. For this purpose, we introduce a novel hybrid evolutionary algorithm able to find high-quality solutions of low conductance for large graphs. Specifically, the proposed algorithm integrates a set of complementary components to jointly ensure its search effectiveness and computational efficiency. We summarize the main contributions of this work as follows.

- First, the hybrid evolutionary algorithm presented in this work takes advantage of population-based global search and local optimization. Specifically, while it adopts a standard crossover operator to generate offspring solutions, the proposed algorithm integrates an innovative progressive local search procedure to cope with the difficulty raised by large graphs (with at least 6×10^4 vertices). To ensure a healthy population of both high diversity and quality, the algorithm uses a mixed technique for population initialization and a proven distance-and-quality based pool updating strategy for population management.
- Second, we perform extensive computational assessments on 60 large-scale real-world benchmark instances (including 50 graphs from the 10th DI-

MACS Implementation Challenge and 10 graphs from the Network Data Repository online, with up to 23 million vertices). We demonstrate the high competitiveness of the proposed algorithm compared to four state-of-the-art algorithms. As an additional assessment, we show an application of using the algorithm to detect community structures in complex networks.

- Third, this work advances the state-of-the-art in terms of effective solving of the general minimum conductance graph partitioning problem. Given that the studied problem has a number of applications, the proposed algorithm can be usefully applied to these practical problems (we will make the code of our algorithm publicly available). Finally, the ideas of some underlying algorithmic components are of general interest and could be advantageously adapted to design effective algorithms for other large graph optimization problems.

The reminder of the paper is organized as follows. In Section 3, we describe the proposed algorithm and its ingredients. In Section 4, we present computational studies and comparisons between the proposed algorithm and state-of-the-art algorithms. An analysis of the key algorithmic components is provided in Section 5. In the last Section, we draw concluding remarks and indicate perspectives for future research.

2 Related work

Table 1
Summary of the key features and technical contributions of the most related studies and the proposed approach.

Reference	Aim	Approach	Main features	Test	Limitations
		Approximation methods			
Cheeger [9] (1969)	Provide a lower bound for the smallest eigenvalue of the Laplacian	Mathematical method	Establish a lower bound in terms of a certain global geometric invariant	No	Specific or particular cases; high computational complexity
Leighton & Rao [22] (1999)	Establish max-flow min-cut theorems for several classes of multicommodity flow problems	Max-flow min-cut algorithm	Implement a $O(logn)$-approximation algorithm for MC-GPP	No	
Arora et al. [2,3] (2004, 2009)	Propose a $O(\sqrt{logn})$-approximation algorithm for MC-GPP	Semidefinite programming	Implement a $O(logn)$-approximation of Leighton and Rao (1999)	No	
Leskovec et al. [24] (2009)	Define and identify clusters or communities by conductance measure	Flow-based, spectral and hierarchical methods	Suggest a detailed and counterintuitive picture of community structures in large social and information networks	Yes	
Spielman et al. [39] (2013)	Determine a good cluster measured by conductance	Nearly linear time local clustering algorithm	Handle massive graphs; find an approximate sparsest cut with nearly optimal balance	Yes	
Zhu et al [46] (2013)	Find well-connected clusters in terms of the conductance	Random-walk based local algorithms	Improve significantly the conductance of cluster where it is well-connected inside	Yes	
		Exact methods			

Hochbaum [18] (2010)	Solve the ratio region problem, a variant of normalized cut in the field of image segmentation	Max-flow min-cut algorithm	Exact solution to the given problem	No	Specific or particular cases; high computational complexity; unpractical for large graphs
Hochbaum [19] (2013)	Solve a discrete relaxation of a family of NP-hard Rayleigh problems	Max-flow min-cut algorithm	Solution to the discrete Rayleigh ratio problem without balance constraint in strongly polynomial time; better solutions than spectral approach	Yes	
		Heuristic and metaheuristic methods			
Lang & Rao [21] (2004)	Improve a graph cut when cut quality is measured by quotient-style metrics such as expansion or conductance	Max-flow min-cut algorithm	Refine the results of Metis graph partitioning heuristic; run in nearly linear time	Yes	Decrease of performance on massive graphs (with millions of vertices)
Andersen & Lang [1] (2008)	Find a larger-than-expected intersection with lower conductance	Sequence and polynomially many applications of max-flow min-cut algorithm	Prove a stronger guarantee of the lowest conductance; improve the quality of cuts without impacting the running time	Yes	
Chalupa [7,8] (2017,2018)	Solve MC-GPP as a pseudo-Boolean optimization problem	Local search and memetic search	Basic search strategies; applied to real-world social networks	Yes	
Lu et al [28] (2019)	Find low conductance solution for large problem instances	Breakout local search (BLS)	Advanced iterated search; applied to large real-world social networks	Yes	
This work	Enrich the toolkit of practical methods for MC-GPP	Hybrid evolutionary algorithm	Fast local optimization; advanced pool diversity management; applied to large and massive networks	Yes	

Existing algorithms for MC-GPP are based on approximation, exact and heuristic approaches. In [9], the first (and weak) approximation algorithm for MC-GPP was presented. Improved approximation algorithms were studied later in [2,3,22]. More recently, motivated by large-scale graph clustering in social and information networks, several effective algorithms with performance guarantee were proposed in [24,39,46]. In [18,19], time-efficient exact algorithms were proposed for a variant of the conductance problem and other ratio problems in the context of image segmentation.

In addition to these approximation and exact approaches, heuristic methods were also studied in order to find high-quality solutions within an acceptable amount of time. One representative method is the Max-flow Quotient-cut Improvement algorithm (MQI) proposed in [21]. This method refines an initial cut (typically given by the Metis graph partitioning heuristic [20]) by solving the well-known max-flow problem. This approach was extended in [1] by solving a sequence of polynomially many minimum cut problems to find a larger-than-expected intersection with lower conductance. Recently, several metaheuristic algorithms using local search and memetic search were studied in [7,8]. Very recently, a stagnation-aware breakout tabu search algorithm (SaBTS) was introduced in [28], which achieved state-of-the-art results on a large test suite composed of small and medium size graphs (with $< 500,000$ vertices). According to the intensive experimental studies reported in [28], the max-flow based MQI algorithm [21] and the SaBTS algorithm [28] significantly dominate the algorithms presented in [7,8]. Table 1 summarizes the main characteristics of these solution methods.

One notices that the best performing algorithms such as MQI and SaBTS are computationally extensive and generally their performance decreases with the size of the problem instances to be solved. This is particularly true for MQI when it is applied to massive graphs with more than 500,000 vertices. In this work, we aim to advance the state-of-the-art of solving MC-GPP by proposing an effective hybrid evolutionary algorithm. Compared to the existing methods, the proposed algorithm distinguishes itself by some original features such as a fast local optimization using a progressive neighborhood and a diversity guarantee strategy based on distance and quality. As we show in Section 4, the proposed algorithm competes very favorably with the state-of-the-art MC-GPP algorithms on the tested large benchmark instances.

3 Memetic algorithm for MC-GPP

3.1 General Outline

Hybrid evolutionary algorithms are known to be a powerful and proven tool for numerous NP-hard problems (e.g., graph partitioning [4,13,43], quadratic assignment [5], graph coloring [30,33], and unconstrained binary quadratic programming [26]) as well as many practical problems (e.g., service placement in fog architectures [15], roadside unit deployment [31] and task scheduling and data assignment on clouds [40]). In this work, we present a hybrid evolutionary algorithm for solving MC-GPP, which relies particularly on the memetic computing framework [32]. The proposed algorithm (denoted by MAMC, see Algorithm 1) is composed of four main components: quality-and-diversity based population initialization, recombination operator, progressive constrained neighborhood tabu search and distance-and-quality based pool updating procedure.

From an initial population *Pop* of p solutions (Algorithm 1, line 1), the best solution s^* among the population is first recorded (line 2). The algorithm then enters into the 'repeat' loop to improve the population (lines 3-18). At each loop, MAMC selects two parent solutions at random and applies the double-point crossover to generate two offspring s^1 and s^2 (lines 4-5). Subsequently, the offspring solutions are improved by the progressive constrained neighborhood tabu search procedure (lines 7 and 13). The improved offspring solutions are used to update the population according to the quality-and-distance based procedure (lines 11 and 17). During the search process, the best solution s^* is updated each time a new best solution is found (lines 8-10, lines 14-16). This process terminates when a given stopping condition is met, which is typically a maximum allowed cut-off time limit.

Algorithm 1 Main framework of the memetic algorithm for the minimum conductance graph partitioning problem (MAMC).

```
Require: Graph $G = (V, E)$, depth of tabu search $d$, size of population $p$.
Ensure: The best partition $s^*$ found during the search.
 1: $Pop = \{s^1, s^2, \ldots, s^p\} \leftarrow Population\_Initialization()$      /* Section 3.2 */
 2: $s^* \leftarrow \arg\min\{\Phi(s^i) : i = 1, 2, \ldots, p\}$
 3: repeat
 4:     Randomly select two parent solutions $s^i$ and $s^j$ from $Pop$
 5:     $(s^1, s^2) \leftarrow Crossover(s^i, s^j)$      /* Section 3.3 */
 6:     /* Trajectory 1: improve $s^1$ with progressive constrained neighborhood tabu
        search */
 7:     $s^1 \leftarrow Tabu\_search(s^1, d)$      /* Section 3.4 */
 8:     if $\Phi(s^1) < \Phi(s^*)$ then
 9:         $s^* \leftarrow s^1$      /* update the best solution found so far */
10:     end if
11:     $Pop = \{s^1, s^2, \ldots, s^p\} \leftarrow Pool\_Updating(s^1, Pop)$      /* Section 3.5 */
12:     /* Trajectory 2: improve $s^2$ by progressive constrained neighborhood tabu
        search */
13:     $s^2 \leftarrow Tabu\_search(s^2, d)$
14:     if $\Phi(s^2) < \Phi(s^*)$ then
15:         $s^* \leftarrow s^2$
16:     end if
17:     $Pop = \{s^1, s^2, \ldots, s^p\} \leftarrow Pool\_Updating(s^2, Pop)$
18: until Stopping condition is satisfied
19: return $s^*$
```

3.2 Quality-and-diversity based population initialization

The proposed MAMC algorithm is characterized by its capacity of maintaining a healthy population with solutions of high diversity and good quality. For the initial population, we adopt the following three-step mixed strategy illustrated by Algorithm 2. First, a seeding partition is generated either randomly or by applying the MQI algorithm [21] with equal probability (lines 2-7). Second, the seeding partition is improved by the progressive constrained neighborhood tabu search of Section 3.4 (line 8). Third, the improved partition is added into the population if it is not already present in the population (lines 9-11). This process is repeated until the population is filled with p different solutions. Given that each solution of the population comes from a seeding solution which is either randomly generated or provided by MQI and then improved by the local optimization procedure, the population is expected to be diverse and of high quality. In Section 5.2, we present an experimental study to show the merit of this mixed initialization strategy compared to two random initialization techniques.

Algorithm 2 The quality-and-diversity based population initialization

Require: Graph $G = (V, E)$, depth of tabu search d, size of population p.
Ensure: The initial population $Pop = \{s^1, s^2, \dots, s^p\}$.

```
 1: $Pop \leftarrow \emptyset$
 2: for $i = 1, \dots, p$ do
 3:     if $rand(0,1) < 0.5$ then
 4:         Construct a random partition $s^0 = (A, \bar{A})$ with $A = \{v\}, \bar{A} = V - \{v\}$
 5:     else
 6:         $s^0 \leftarrow MQI()$ /* Create $s^0$ by applying the max-flow algorithm MQI [21] */
 7:     end if
 8:     $s^0 \leftarrow Tabu\_Search(s^0, d)$                                /* Section 3.4 */
 9:     if $s^0$ is different from all solutions in $Pop$ then
10:         $Pop \leftarrow \{s^0\} \cup Pop$
11:     end if
12: end for
13: return $Pop$
```

3.3 Recombination operator and parent selection

The crossover operator is used to generate new solutions from parent solutions. For MC-GPP, we experimented three standard crossovers and one problem-specific crossover. For our discussion, we note that a partition $s = (A, \bar{A})$ can be conveniently coded by a binary string of length n (n is the number of the vertices in the given graph). Indeed, we can use a binary variable to represent a vertex and set it to 1 if the vertex belongs to A and set it to 0 if the vertex belongs to $\bar{A}$. As a result, conventional crossovers that typically operate on binary strings are directly applicable. Below, we describe the crossovers we tested for solving MC-GPP.

- Single-point crossover: A crossover point on both parents is chosen randomly and then the two bit strings of the parents to the right of the crossover point are swapped to generate two offspring solutions.
- Double-point crossover: Two different crossover points on both parents are chosen randomly and the two bit strings between the two crossover points are swapped to generate two offspring solutions (see Fig. 2 for an example).
- Uniform crossover: Each bit of the offspring is randomly chosen from either parent with equal probability.
- Cut-edge-preserving crossover: In this problem-specific crossover, the cut-edges shared by both parents are preserved in the offspring. The vertices that are not endpoints of these shared cut-edges are assigned randomly the value of 1 or 0.

According to our experiments, the double-point crossover performs globally the best compared to the other crossovers. We adopt thus the double-point

$$\begin{tikzcd}
\mathit{Parent}_1 : 11|\mathbf{10100}|1000 \arrow[dr] & \mathit{Child}\ s^1 : 11|\mathbf{00101}|1000 \\
\mathit{Parent}_2 : 00|\mathbf{00101}|0101 \arrow[ur] & \mathit{Child}\ s^2 : 00|\mathbf{10100}|0101
\end{tikzcd}$$

Fig. 2. The double-point crossover

crossover in our MAMC algorithm.

As to parent selection, we adopt the simple random selection instead of other mechanisms such as roulette-wheel selection and tournament selection. This is justified by the fact that thanks to the distance-and-quality pool updating procedure of Section 3.5 and the powerful local optimization procedure of Section 3.4, we make sure that the population is composed of solutions which are pair-wisely distanced and of high quality. As such, any pair of selected solutions are guaranteed to be well separated and have a high fitness.

3.4 Progressive constrained neighborhood tabu search

It is well known that hybrid evolutionary algorithms need an effective local optimization procedure to ensure search intensification [17]. In [28], it is shown that both the max-flow based MQI algorithm and the stagnation-aware breakout tabu search algorithm (SaBTS) perform well on the tested graphs with up to 500,000 vertices. This is particularly true, when they are used to refine an initial solution of good quality provided by the Metis graph partitioning tool [20]. As a result, both MQI and SaBTS could be used as our local optimization procedure in principle. On the other hand, these algorithms become time-consuming when we handle large and massive graphs, making them less interesting and unsuitable for the purpose of this work where the studied graphs have up to 23 million vertices. To be able to cope with such large graphs, we created the progressive constrained neighborhood tabu search algorithm (PCNTS) that is presented below.

Basically, our PCNTS procedure uses the 'Relocate' operator to explore "critical" vertices [28]. Specifically, let $s = (A, \bar{A})$ be the incumbent solution and v a vertex, then $\mathit{Relocate}(v)$ transforms s to a new (neighbor) solution s' (denoted by $s' = s \oplus \mathit{Relocate}(v)$) by displacing vertex v from its current set A or $\bar{A}$ to the complement set. For the purpose of conductance minimization, it is unnecessary to consider a vertex for relocation if the vertex is not the endpoint of a cut edge [28]. As a result, we identify the set $CV(s)$ of candidate vertices for relocation as $CV(s) = \{v \in V : v \text{ is the endpoint of a cut edge of } s\}$. Note that even if the set $CV(s)$ is much smaller than the set of vertices V, examining all critical vertices at each iteration of the algorithm (this is what SaBTS of [28] does) can still become very expensive for large and massive graphs.

Algorithm 3 Progressive constrained neighborhood tabu search (PCNTS).

```
Require: Graph $G = (V, E)$, current solution $s$, depth of tabu search $d$.
Ensure: The best solution found $s^b$.
 1: $s^b \leftarrow s$          /* record the best solution found during the current TS run */
 2: $H \leftarrow \emptyset$
 3: $\beta \leftarrow 0$          /* counter of consecutive non-improving iterations w.r.t. $s^b$ */
 4: $ev \leftarrow 1$          /* $ev$ is the number of evaluating vertices in $\mathcal{N}_C(s)$ */
 5: Create the set $CV(s, ev)$ of critical vertices
 6: while $\beta < d$ do
 7:     Select a best eligible vertex $v$ in $CV(s, ev)$
 8:     $s \leftarrow s \oplus Relocate(v)$
 9:     Update the set of critical vertices $CV(s, ev)$
10:     Update tabu list $H[v]$ with tabu tenure $tt$
11:     if $\Phi(s) < \Phi(s^b)$ then
12:         $s^b \leftarrow s$
13:         $\beta \leftarrow 0$, $ev \leftarrow 1$
14:     else
15:         $\beta \leftarrow \beta + 1$, $ev \leftarrow ev + 1$
16:     end if
17:     if $ev > |CV(s)|$ then
18:         $ev = 1$
19:     end if
20: end while
21: return $s^b$
```

This is especially true when the given graph is dense, because in this case, any partition will have a high number of cut edges, thus implying many critical vertices. Moreover, as discussed in [7,28], unlike the conventional graph k-way partitioning problem where examining all 'Relocate' neighbor solutions can be performed in $O(1)$, no technique is known to ensure such an efficiency for performing 'Relocate' for MC-GPP. This implies that the cost of examining neighbor solutions is proportionally correlated to the number of critical vertices and becomes prohibitive for large graphs.

To cope with this difficulty, we devise the following *progressive and elastic neighborhood exploration strategy* which dynamically increases or decreases the number of examined critical vertices according to a well-defined condition (see Algorithm 3). Specifically, we start with one critical vertex (indicated by $ev = 1$, ev is a counter indicating the number of the critical vertices to be examined). If relocating this vertex at the current iteration does not lead to a solution better than the recorded local best solution (s^b), we increase ev by one, implying that during the next iteration, one more critical vertex will be examined. As a result, as long as no improved local best solution is found, a still larger neighborhood exploration is enabled by increasing ev (i.e., the number of candidate critical vertices) during the next iteration. During the search, ev is reset to one as soon as the recorded local best solution (s^b) is updated or ev reaches its upper limit $|CV(s)|$. The progressive constrained neighborhood tabu search procedure thus explores an elastic neighborhood $\mathcal{N}_{AC}(s)$ whose size is dynamically adjusted by the counter ev according to the search state.

$$\mathcal{N}_{AC}(s) = \{s' = s \oplus \mathit{Relocate}(v) :\ v \in CV(s, ev), 1 \leq ev \leq |CV(s)|\} \tag{13}$$

where $CV(s, ev) \subset CV(s)$ is the set of critical vertices to be examined.

Thus, by examining the neighborhood $\mathcal{N}_{AC}(s)$ induced by $CV(s, ev)$ instead of the whole neighborhood induced by $CV(s)$, the PCNTS procedure increases its computational efficiency considerably.

For a given neighbor solution s' in the neighborhood $\mathcal{N}_{AC}(s)$, it is necessary to quantify the conductance variation (called move gain) $\delta(v) = \Phi(s') - \Phi(s)$ with $\Phi(s)$ already known. This can be performed in $O(1)$ time by simply calculating $|cut(s')|, vol(A'), vol(\bar{A}')$ as follows [7,28].

$$|cut(s')| = |cut(s)| + deg_A(v) - deg_{\bar{A}}(v) \tag{14}$$
$$vol(A') = vol(A) - deg(v) \tag{15}$$
$$vol(\bar{A}') = vol(\bar{A}) + deg(v) \tag{16}$$

where $s' = \{A', \bar{A}'\}$ with $A' = A \setminus \{v\}$, $\bar{A}' = \bar{A} \cup \{v\}$ is the neighbor solution after relocating v from A to $\bar{A}$ of the solution $s = (A, \bar{A})$.

To explore the progressive constrained neighborhood $\mathcal{N}_{AC}(s)$, the PCNTS procedure first identifies the set $CV(s, ev)$ with one critical vertex (Algorithm 3, line 5). This can be achieved in $O(1)$ time. Then at each iteration, PCNTS selects one best vertex v in terms of move gain achieved in $O(|CV(s, ev)|)$ time among the *eligible* critical vertices (ties are broken randomly) and relocates v to obtain a neighbor solution. A vertex qualifies as eligible if it is not forbidden by the tabu list (see below). Note that if a vertex leading to a solution better than the recorded best solution s^b is always selected, even if it is forbidden by the tabu list (this is called the aspiration criterion in tabu search).

It is possible that the above search procedure revisits a previously encountered solution, thus leading to search cycling. To prevent this, we use a memory H (called tabu list) to record each displaced vertex and forbid the vertex to be relocated again during a specific number tt of iterations (called tabu tenure). Thus, each time a vertex $v \in CV$ is relocated to generate a neighbor solution, the PCNTS procedure adds v in H (an integer vector in our case) and will ignore this vertex for next $tt(v)$ iterations. In practice, when v is relocated,

$H[v]$ is set to $iter + tt$ where $iter$ represents the current number of iterations. Then during the next iterations, if $iter < H[v]$, v is forbidden by the tabu list; otherwise, v is not prohibited by the tabu list.

For the tabu tenure, we adopt the dynamic technique introduced in [13] which has proven to be quite robust and effective for graph partitioning problems [28,43]. This technique varies tt with a periodic step function F defined over the current iteration number $iter$. For each period, it takes $p = 15$ successive values (also called intervals). The values are separated by the interval margins which are defined by $x_1 = 1$, and $x_{i+1} = x_i + 100$. tt equals $F(iter)$, which is given by $(y_i)_{i=1,2,\ldots,15} = \alpha \times (10, 20, 10, 40, 10, 20, 10, 80, 10, 20, 10, 40, 10, 20, 10)$ (α is a parameter). Therefore, tt equals $10 \times \alpha$ between iterations 1 and 100, $20 \times \alpha$ between iterations 101 and 200, etc. Our experiments confirmed that this technique makes the algorithm quite robust across the tested instances and avoids the difficulties of manually tuning the tabu tenure.

3.5 Distance-and-quality based pool updating procedure

Population diversity is critical to avoid a premature convergence of an evolutionary algorithm. To maintain a healthy population of our algorithm, we employ a diversification preserving strategy [26] to update the population with each new offspring solution. This strategy considers not only the quality of each offspring solution, but also its distance to the existing solutions of the population.

Following [33], we use the well-known set-theoretic partition distance [16] to measure the distance d_{ab} between two solutions s^a and s^b. The partition distance is the minimum number of one-move steps necessary to transform one solution to another solution. For the given population Pop, we calculate $D_{i,Pop} = min\{d_{ij} | s^j \in Pop,\ j \neq i, i, j = 1, \ldots, |Pop|\}$.

Then, we determine the 'worst' solution in the population, according to the following goodness score function which considers both quality and distance:

$$g(s^i, Pop) = \gamma \tilde{A}(\Phi(s^i)) + (1 - \gamma)\tilde{A}(D_{i,pop}) \tag{17}$$

where γ is a parameter set to 0.6 according to [26] and $\Phi(s^i)$ is the objective function value (conductance) of solution s^i and $\tilde{A}(\cdot)$ is the normalized function:

$$\tilde{A}(y) = \frac{y - y_{min}}{y_{max} - y_{min} + 1} \tag{18}$$

Algorithm 4 Distance-and-quality based updating procedure

Require: Graph $G = (V, E)$, offspring solution s^0, size of population p, population $Pop = \{s^1, s^2, \dots, s^p\}$.

Ensure: The updated population $Pop = \{s^1, s^2, \dots, s^p\}$.

```
 1: $Pop' \leftarrow Pop \cup \{s^0\}$                /* Create a temporary population $Pop'$ */
 2: for $i = 0, \dots, p$ do
 3:     Calculate the distance $D_{i,Pop'}$ between $s^i$ and $Pop'$
 4:     Calculate the goodness score $g(s^i, Pop')$ of $s^i$ according to Eq. (17)
 5: end for
 6: $s^w \leftarrow$ arg max$\{g(s^i, Pop') : i = 0, 1, \dots, p\}$ /* $s^w$ is the worst solution $s^w$ in
    $Pop'$ */
 7: if $s^w \neq s^0$ then
 8:     $Pop \leftarrow Pop \setminus \{s^w\} \cup \{s^0\}$          /* Replace $s^w$ with $s^0$ in $Pop$ */
 9: else
10:     if $rand(0, 1) < 0.5$ then
11:         $s^{sw} \leftarrow$ arg max$\{g(s^i, Pop') : i = 0, 1, \dots, p, s^i \neq s^w\}$ /* Identify the second
            worst solution $s^{sw}$ in $Pop'$ */
12:         $Pop \leftarrow Pop \setminus \{s^{sw}\} \cup \{s^0\}$        /*Replace $s^{sw}$ with $s^0$ in $Pop$ */
13:     end if
14: end if
15: return $Pop$
```

where y_{min} and y_{max} are respectively the minimum and maximum of y in the population *Pop*. "+1" is used to avoid the possibility of a 0 denominator.

Based on the above goodness score function, the population *Pop* is updated with the given offspring solution s^0 according to the following procedure (Algorithm 4). The new offspring s^0 is first inserted into $Pop = \{s^1, s^2, \dots, s^p\}$ to create a temporary population $Pop' = \{s^0, s^1, \dots, s^p\}$ (line 1). Then for each solution of Pop', its distance to Pop' and goodness score are calculated (lines 2-5). After that the worst solution s^w in Pop' is identified according to the score function (17) (line 6). Finally, if s^w is different from the offspring s^0, s^0 replaces the worst solution s^w in the population (line 7-8). Otherwise, s^0 replaces the second worst solution s^{sw} is replaced by s^0 with a probability of 0.5 (lines 10-13).

4 Computational studies

In this section, we first assess the proposed MAMC algorithm for solving MC-GPP based on two sets of 60 benchmark graphs from various applications, and then apply MAMC to detect communities of 3 real-world complex networks.

4.1 Benchmark instances

The two sets of benchmarks include 60 large and massive graphs which are from two sources and have 60,005 to 23,947,347 vertices.

The 10th DIMACS Implementation Challenge Benchmark.This dataset contains 50 graphs which are dedicated to two related problems of graph partitioning and graph clustering. These graphs belong to 6 families: Clustering instances, Delaunay graphs, Redistricting, Walshaw's graph partitioning archive, Co-author and citation networks, and Sparse matrices. They are available at: `https://www.cc.gatech.edu/dimacs10/downloads.shtml`.

The Network Data Repository online. This dataset contains 10 massive real-world network graphs [34]. This dataset can be found at: `http://networkrepository.com/index.php`.

Additionally, we also test our approach for community detection on 3 popular real-world complex networks (Zachary's Karate Club, College Football Network and Bottlenose Dolphin Social Network) available at: `http://www-personal.umich.edu/~mejn/netdata/`.

4.2 Experimental setting

The proposed MAMC algorithm was coded in C++ [1] and compiled using GNU g++ 6.3.0 compiler with the "-O3" flag. The experiments were conducted on a computer running Ubuntu Linux 16.04, using 6 cores of AMD Opteron 4184 CPU @ 2.80GHz and 32GByte RAM.

To evaluate our results, we adopt as our references the following four state-of-the-art algorithms in the literature.

- Metis [20]: This is a general and popular graph partitioning package which has been used to generate partitions in several studies on MC-GPP [1,21,24]. As shown in [28], even if Metis does not directly optimize the conductance criterion (it minimizes the number of cut edges), it can produce partitions of relatively low conductance within a very short computation time. In our study, Metis is used to serves as a baseline reference, as well as to create initial partitions of the MQI and SaBTS methods (see below). For this work, we used the latest release Metis 5.1.0 available at: `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`.

[1] The code of our MAMC algorithm will be available at: `http://www.info.univ-angers.fr/~hao/mamc.html`

Table 2
Settings of parameters.

Parameter	Section	Description	Value
p	3.2	population size	20
α	3.4	tabu tenure management factor	10
d	3.4	depth of tabu search	6000

- MQI [21]: This is a max-flow quotient-cut improvement algorithm which refines a given initial partition. In previous studies and this work, MQI starts with a partition provided by the fast Metis tool. To our knowledge, MQI is one of the best algorithms for MC-GPP. We used the latest implementation of MQI available at: `https://github.com/kfoynt/LocalGraphClustering`.
- SaBTS [28]: This is the most recent metaheuristic algorithm for MC-GPP, which combines a dedicated tabu search procedure and a self-adaptive perturbation procedure. The intensive experiments reported in [28] show that SaBTS is able to consistently improve a partition provided by Metis and can also reduce the conductance of partitions given by MQI (see below). The code of SaBTS is available at: `http://www.info.univ-angers.fr/~hao/mcgpp.html`.
- MQI+SaBTS [28]: This hybrid approach uses SaBTS to refine a partition produced by MQI. Thanks to the combination of these two powerful approaches (MQI and SaBTS), MQI+SaBTS performs the best compared to other existing approaches.

The proposed MAMC algorithm requires 3 parameters: population size p, tabu tenure management factor α and depth of tabu search d. Following previous studies [13,43], we adopted a small population size and set $p = 20$. For α and d, they were fixed according to the analysis reported in Section 5.1. For our experiments, we consistently used the parameter setting shown in Table 2 to run our MAMC algorithm to solve all instances. This parameter setting can also be considered to be the default setting of MAMC.

For a fair comparison, we run, on the same computer, our MAMC algorithm and the above reference algorithms with their respective default parameter setting under the same time limit of 60 minutes per run and per instance.

4.3 Computational results and comparison on the benchmark instances

In this section, we report the results of the proposed MAMC algorithm as well as the four reference algorithms (Metis [20], MQI [21], SaBTS [28] and MQI+SaBTS [28]) on the 60 benchmark instances. It is worth noting that MQI, SaBTS and MQI+SaBTS all start from an initial partition provided by

Metis. Tables 3 and 4 provide the detailed results of the compared algorithms, while Table 5 shows a summary.

In Tables 3 and 4, columns 1-2 indicate the name (Graph) and the number of vertices ($|V|$) for each instance. The remaining columns show the results of Metis, MQI, SaBTS, MQI+SaBTS and MAMC according to the following performance indicators: the best conductance (Φ_{best}) found among 20 runs, the average conductance (Φ_{avg}), the success rate (*hit*) over 20 runs to reach Φ_{best}, the average CPU time in seconds ($t(s)$) of 20 runs to attain the best results, and the standard deviation (σ) of Φ_{best}. It is worth noting that computation times are provided only for indicative purposes, since it is not meaningful to compare two computation times if the corresponding algorithms lead to solutions of different quality.

In Table 5, column 1 indicates the pairs of compared algorithms. Column 2 gives the total number of instances (#Instance). Column 3 shows the quality indicators in terms of the best and average conductance (Φ_{best} and Φ_{avg}). Columns 4-6 count the number of instances on which MAMC achieves a better, equal or worse result compared to each reference algorithm (#Wins, #Ties and #Losses). Column 7 reports the p-value from the non-parametric Wilcoxon signed-rank test with a confidence level of 99%.

The results of Tables 3–5 show that MAMC performs remarkably well on all 60 benchmark instances. Compared to Metis and MQI and in terms of the main performance indicators Φ_{best} (Φ_{avg}), MAMC finds 54 (60) better and 6 (0) equal results with respect to Metis, and 27 (60) better and 33 (0) equal results with respect to MQI. MAMC also competes very favorably with SaBTS and MQI+SaBTS in terms of Φ_{best} (Φ_{avg}) with 53 (58) wins, 7 (1) ties and 0 (1) losses compared to SaBTS, and 27 (58) wins, 32 (2) ties and 1 (0) losses compared to MQI+SaBTS. The small p-values ($p\text{-}value \ll 0.01$) from the Wilcoxon signed-rank further confirm the dominance of MAMC over the reference algorithm in terms of Φ_{best} and Φ_{avg}.

Moreover, in terms of the other performance indicators, we observe that to reach the same Φ_{best} value, MAMC has always a higher success rate and a shorter computation time. In many cases, MAMC is even able to find a better solution with a higher hit and a shorter time.

To complete these results, we additionally provide a performance assessment of the algorithms in a visual way by using a generic benchmarking tool called *performance profiles* [11] (the reader is referred to [11] for details). Performance profiles enable a rigorous comparison of different algorithms over a large set of benchmark instances with regard to a specific performance metric (in our case, Φ_{best} and Φ_{avg} respectively). Basically, to compare a set of algorithms $\mathcal{S}$ over a set of problems $\mathcal{P}$, we define the performance ratio by $r_{s,p} = \frac{\Phi_{s,p}}{min\{\Phi_{s,p}: s \in \mathcal{S}\}}$. If an algorithm s does not solve a problem p, then we simply set $r_{s,p} = +\infty$. Thus, the performance function of an algorithm s is given by $\mathcal{P}_s(\tau) = \frac{|\{p\in\mathcal{P} \mid r_{s,p}\leq\tau\}|}{|\mathcal{P}|}$. The value $\mathcal{P}_s(\tau)$ computes the fraction of problems algorithm s can solve with at most τ many times the cost of the best algorithm. $\mathcal{P}_s(1)$ corresponds to the number of problems that algorithm s solved faster than, or as fast as the other algorithms in $\mathcal{S}$. The value $\mathcal{P}_s(r_f)$, for a large enough r_f, corresponds to the maximum number of problems that algorithm s has solved. The quantities $\mathcal{P}_s(1)$ and $\mathcal{P}_s(r_f)$ are called efficiency and robustness of s respectively.

Fig. 3 shows the performance profiles of our MAMC algorithm as well as the reference algorithms which have been drawn with the software `perprof-py` [38]. From the figure, we observe that MAMC has a very good performance, surpassing the four reference algorithms in terms of conductance value. Indeed, MAMC has the highest value of $\mathcal{P}_s(1)$ among the compared algorithms, meaning that MAMC can quickly find the lowest conductance for the tested instances. Besides, MAMC also attains a good robustness by quickly solving all the instances (corresponding to the fact that MAMC arrives at $\mathcal{P}_s(r_f)$ first).

This experiment thus demonstrates the competitiveness of the proposed MAMC algorithm for solving MC-GPP compared to the 4 state-of-the-art reference methods.

Table 3
Detailed computational results of MAMC with two state-of-the-art algorithms Metis [20], MQI [21] on 50 large graphs from the 10th DIMACS Challenge and 10 large graphs from the Network Data Repository online. The best of the Φ_{best} values for each instance is highlighted in boldface, while the best of the Φ_{avg} values for each instance is indicated in italic.

Instance		Metis [20]					MQI [21]					**MAMC**				
Graph	$\|V\|$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	σ	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	σ	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	σ
preferential Attachment	100000	0.31136208	0.33179699	1/20	0	8.45e-03	0.31132997	0.33039853	1/20	4	7.91e-03	**0.28717318**	*0.28804940*	1/20	2886	6.17e-04
smallworld	100000	0.11353648	0.11625592	1/20	0	1.72e-03	0.11322141	0.11604861	1/20	9	1.72e-03	**0.09948259**	0.10321531	1/20	2739	1.27e-03
cnr-2000	325557	0.00014499	0.00045316	1/20	0	2.30e-04	**0.00000156**	0.00000649	12/20	24	6.76e-06	**0.00000156**	*0.00000156*	20/20	1	0
eu-2005	862664	0.00180465	0.00297210	1/20	4	8.45e-04	**0.00004930**	0.00064963	5/20	224	8.75e-04	**0.00004930**	*0.00004930*	20/20	11	0
road_central	14081816	**0.00001000**	0.00001399	1/20	38	1.52e-06	**0.00001000**	0.00001399	1/20	3620	1.52e-06	**0.00001000**	*0.00001128*	9/20	272	1.21e-06
road_usa	23947347	**0.00000584**	0.00000743	1/20	46	8.70e-07	**0.00000584**	0.00000743	1/20	3625	8.70e-07	**0.00000584**	*0.00000606*	11/20	493	2.80e-07
delaunay_n16	65536	0.00245485	0.00255656	1/20	0	5.08e-05	0.00233083	0.00239444	1/20	37	4.40e-05	**0.00226376**	*0.00229371*	1/20	1292	2.44e-05
delaunay_n17	131072	0.00176260	0.00180880	1/20	0	3.15e-05	0.00161703	0.00166800	1/20	109	1.91e-05	**0.00156857**	*0.00161819*	1/20	2000	2.22e-05
delaunay_n18	262144	0.00122712	0.00128721	1/20	0	2.50e-05	0.00113731	0.00117238	1/20	251	1.89e-05	**0.00113332**	*0.00114381*	1/20	762	8.03e-06
delaunay_n19	524288	0.00087932	0.00090623	1/20	1	1.86e-05	0.00078966	0.00081794	1/20	1807	1.16e-05	**0.00078693**	*0.00080017*	1/20	729	9.73e-06
delaunay_n20	1048576	0.00062086	0.00064508	1/20	1	1.26e-05	0.00055891	0.00057390	1/20	3601	9.59e-06	0.00055891	*0.00056128*	9/20	27	3.50e-06
delaunay_n21	2097152	0.00043742	0.00045557	1/20	3	8.39e-06	**0.00039293**	0.00041813	1/20	3603	3.01e-05	**0.00039293**	*0.00039397*	8/20	45	1.35e-06
delaunay_n22	4194304	**0.00031352**	0.00032532	1/20	8	6.42e-06	**0.00031352**	0.00032532	1/20	3605	6.42e-06	**0.00031352**	*0.00031504*	13/20	286	2.61e-06
delaunay_n23	8388608	**0.00022443**	0.00023220	1/20	16	4.08e-06	**0.00022443**	0.00023220	1/20	3611	4.08e-06	**0.00022443**	*0.00022552*	8/20	526	1.30e-06
delaunay_n24	16777216	**0.00016180**	0.00016509	1/20	31	2.32e-06	**0.00016180**	0.00016509	1/20	3621	2.32e-06	**0.00016180**	*0.00016208*	11/20	1256	3.20e-07
co2010	201062	0.00067974	0.00073333	1/20	0	3.29e-05	0.00053701	0.00061163	1/20	116	4.98e-05	**0.00053442**	*0.00053453*	19/20	564	4.80e-07
la2010	204447	0.00037015	0.00042879	1/20	0	5.70e-05	**0.00017789**	0.00028106	3/20	150	7.38e-05	**0.00017789**	*0.00017789*	20/20	4	0
ia2010	216007	0.00067628	0.00072474	1/20	0	2.45e-05	0.00060103	0.00062588	1/20	192	1.88e-05	**0.00059147**	*0.00060046*	1/20	1239	4.62e-06
ks2010	238600	0.00058509	0.00063764	1/20	0	2.36e-05	0.00051471	0.00054122	1/20	226	1.58e-05	**0.00051470**	*0.00051875*	10/20	980	4.54e-06
tn2010	240116	0.00043558	0.00048995	1/20	0	3.43e-05	**0.00033671**	0.00039117	7/20	257	4.45e-05	**0.00033671**	*0.00033671*	20/20	4	0
az2010	241666	0.00057028	0.00063012	1/20	0	3.36e-05	0.00044358	0.00051400	1/20	164	4.14e-05	**0.00043839**	*0.00045183*	1/20	94	1.90e-05
al2010	252266	0.00059056	0.00065367	1/20	0	3.57e-05	0.00051345	0.00054504	1/20	221	2.70e-05	**0.00050732**	*0.00051154*	2/20	1295	4.01e-06
wi2010	253096	0.00053361	0.00063087	1/20	0	7.26e-05	0.00049104	0.00053608	1/20	225	4.73e-05	**0.00048247**	*0.00048581*	3/20	1374	4.78e-06
mn2010	259777	0.00061880	0.00066681	1/20	0	2.28e-05	0.00053576	0.00055713	1/20	251	1.35e-05	**0.00053491**	*0.00053926*	1/20	828	3.59e-06
in2010	267071	0.00054622	0.00060620	1/20	0	2.93e-05	**0.00045199**	0.00047775	7/20	233	3.19e-05	**0.00045199**	*0.00045199*	20/20	2	0
ok2010	269118	0.00054795	0.00060312	1/20	0	3.21e-05	0.00046031	0.00050643	1/20	236	2.15e-05	**0.00046030**	*0.00046677*	2/20	501	9.09e-06
va2010	285762	0.00060428	0.00066761	1/20	0	3.86e-05	**0.00050629**	0.00054958	4/20	287	3.43e-05	**0.00050629**	*0.00050810*	17/20	3	6.23e-06
nc2010	288987	0.00033192	0.00036179	1/20	0	2.15e-05	0.00029130	0.00030411	1/20	267	8.45e-06	**0.00029079**	*0.00029215*	1/20	1218	2.44e-06
ga2010	291086	0.00041835	0.00047506	1/20	0	4.09e-05	0.00036488	0.00039249	1/20	224	2.28e-05	**0.00035935**	*0.00036353*	1/20	1083	2.93e-06
mi2010	329885	0.00050360	0.00055231	1/20	0	2.58e-05	**0.00009247**	0.00022270	12/20	217	1.77e-04	**0.00009247**	*0.00009247*	20/20	1	0

Table 3: Continued

Instance		Metis [20]					MQI [21]					**MAMC**				
Graph	$\|V\|$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	σ	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	σ	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	σ
mo2010	343565	0.00055795	0.00062254	1/20	0	4.58e-05	**0.00044010**	0.00051012	1/20	310	4.15e-05	**0.00044010**	*0.00044334*	13/20	481	1.25e-05
oh2010	365344	0.00055951	0.00060063	1/20	0	2.17e-05	0.00047052	0.00050917	1/20	358	2.27e-05	**0.00045030**	*0.00047809*	1/20	312	9.88e-06
pa2010	421545	0.00034314	0.00039015	1/20	0	3.87e-05	**0.00026616**	0.00028228	10/20	398	2.52e-05	**0.00026616**	*0.00026616*	20/20	2	0
il2010	451554	0.00029503	0.00031679	1/20	0	1.61e-05	0.00025204	0.00026746	1/20	436	9.26e-06	**0.00024997**	*0.00025260*	2/20	515	1.74e-06
tx2010	914231	0.00028327	0.00032283	1/20	1	1.99e-05	0.00024258	0.00025289	1/20	2386	9.50e-06	**0.00024221**	*0.00024274*	2/20	176	4.60e-07
t60k	60005	0.00088178	0.00103187	1/20	0	8.52e-05	**0.00069385**	0.00073514	16/20	45	9.66e-05	**0.00069385**	*0.00069385*	20/20	1	0
wing	62032	0.00709383	0.00734335	1/20	0	1.52e-04	0.00668630	0.00690103	1/20	26	1.37e-04	**0.00654906**	*0.00662078*	1/20	1953	6.69e-05
brack2	62631	0.00199422	0.00207331	1/20	0	4.28e-05	**0.00185332**	0.00185425	19/20	36	4.05e-06	**0.00185332**	*0.00185332*	20/20	10	0
finan512	74752	**0.00062040**	0.00062041	8/20	0	1.00e-08	**0.00062040**	0.00062041	8/20	15	1.00e-08	**0.00062040**	*0.00062040*	20/20	3	0
fe_tooth	78136	0.00908187	0.00955851	1/20	0	2.35e-04	0.00878706	0.00903448	1/20	37	2.30e-04	**0.00843147**	*0.00846699*	1/20	2032	8.13e-05
fe_rotor	99617	0.00323856	0.00331810	1/20	0	4.08e-05	0.00307533	0.00314188	1/20	50	1.80e-05	**0.00302488**	*0.00303411*	14/20	542	2.59e-05
598a	110971	0.00332544	0.00336420	1/20	0	3.70e-05	0.00325673	0.00327184	1/20	80	1.50e-05	**0.00323209**	*0.00323816*	2/20	2145	5.02e-06
fe_ocean	143437	0.00086985	0.00120139	1/20	0	1.72e-04	**0.00078223**	0.00090825	3/20	65	1.50e-04	**0.00078223**	*0.00078223*	20/20	47	0
144	144649	0.00620481	0.00636608	1/20	0	7.57e-05	0.00611994	0.00622039	1/20	99	6.47e-05	**0.00605305**	*0.00607374*	1/20	2690	9.00e-06
wave	156317	0.00853325	0.00865172	1/20	0	1.06e-04	0.00830287	0.00845742	1/20	81	1.20e-04	**0.00819101**	*0.00823630*	1/20	2867	2.25e-05
m14b	214765	0.00233434	0.00239063	1/20	0	3.65e-05	0.00230381	0.00232866	1/20	201	1.90e-05	**0.00228466**	*0.00229867*	1/20	1865	5.47e-06
auto	448695	0.00314960	0.00321218	1/20	1	4.28e-05	**0.00298571**	0.00301454	3/20	412	6.31e-05	**0.00298571**	*0.00298579*	17/20	476	2.00e-07
coAuthorsDBLP	299067	0.05056661	0.05194195	1/20	0	9.04e-04	**0.00411522**	0.00494913	2/20	29	5.22e-04	**0.00411522**	*0.00411522*	20/20	10	0
thermal2	1227087	0.00027475	0.00028069	1/20	1	3.99e-06	**0.00025296**	0.00025687	1/20	3601	6.42e-06	**0.00025296**	*0.00025339*	10/20	223	4.90e-07
G3_circuit	1585478	0.00040859	0.00044443	1/20	1	1.61e-05	**0.00036328**	0.00038076	1/20	3595	1.51e-05	**0.00036328**	*0.00036393*	8/20	335	1.01e-06
ca-coauthors-dblp	540486	0.04069920	0.04310426	1/20	4	1.54e-03	**0.00226757**	0.00244841	15/20	221	3.14e-04	**0.00226757**	*0.00226757*	20/20	5	0
inf-roadNet-PA	1087562	0.00011467	0.00012353	1/20	1	6.39e-06	**0.00008221**	0.00008789	2/20	3593	7.57e-06	**0.00008221**	*0.00008230*	15/20	4	1.60e-07
sc-nasasrb	54870	0.00338961	0.00356608	1/20	0	8.29e-05	**0.00325489**	0.00335273	2/20	89	5.86e-05	**0.00325489**	*0.00325489*	20/20	14	0
sc-pkustk13	94893	0.00930995	0.00973785	1/20	0	2.79e-04	0.00921128	0.00952533	1/20	192	2.56e-04	**0.00905805**	*0.00906634*	15/20	1729	2.86e-05
soc-gowalla	196591	0.06688173	0.06912221	1/20	0	1.05e-03	**0.01234567**	0.01247394	16/20	15	2.57e-04	**0.01234567**	*0.01234567*	20/20	0	0
soc-twitter-follows	404719	0.08818687	0.09068638	1/20	1	1.29e-03	**0.00540540**	0.01522024	1/20	19	6.26e-03	**0.00540540**	*0.00540540*	20/20	38	0
soc-youtube	495957	0.07416606	0.07657564	1/20	2	1.52e-03	**0.00892857**	0.01134011	12/20	58	3.25e-03	**0.00892857**	*0.00892857*	20/20	2	0
soc-flickr	513969	0.07132059	0.11153507	1/20	3	3.28e-02	**0.00444444**	0.00565198	13/20	43	1.66e-03	**0.00444444**	*0.00444444*	20/20	1	0
soc-FourSquare	639014	0.34789798	0.34987880	1/20	3	1.15e-03	**0.26530612**	0.26564625	15/20	40	5.89e-04	**0.26530612**	*0.26530612*	20/20	4	0
web-arabic-2005	163598	0.00000556	0.00003389	1/20	0	9.10e-05	**0.00000306**	0.00000323	10/20	28	5.40e-07	**0.00000306**	*0.00000306*	20/20	1	0

Table 4
Detailed computational results of MAMC with two reference algorithms SaBTS and MQI+SaBTS [28] on 50 large graphs from the 10th DIMACS Challenge and 10 large graphs from the Network Data Repository online. The best of the Φ_{best} values for each instance is highlighted in boldface, while the best of the Φ_{avg} values for each instance is indicated in italic.

Instance		SaBTS [28]					MQI+SaBTS [28]					**MAMC**				
Graph	$\|V\|$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	σ	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	σ	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	σ
preferential Attachment	100000	0.29468142	0.29575422	1/20	1790	7.22e-04	0.29457883	0.29584725	1/20	2264	1.04e-03	**0.28717318**	*0.28804940*	1/20	2886	6.17e-04
smallworld	100000	0.10048440	*0.10143134*	1/20	3191	6.79e-04	0.10105860	0.10387462	1/20	3181	1.99e-03	**0.09948259**	0.10321531	1/20	2739	1.27e-03
cnr-2000	325557	0.00014499	0.00045286	1/20	179	2.30e-04	**0.00000156**	0.00000649	12/20	0	7.00e-06	**0.00000156**	*0.00000156*	20/20	1	0
eu-2005	862664	0.00180465	0.00296178	1/20	118	8.54e-04	**0.00004930**	0.00064963	5/20	1219	8.75e-04	**0.00004930**	*0.00004930*	20/20	11	0
road_central	14081816	**0.00001000**	0.00001399	1/20	0	2.00e-06	**0.00001000**	0.00001399	1/20	3240	1.52e-06	**0.00001000**	*0.00001128*	9/20	272	1.21e-06
road_usa	23947347	**0.00000584**	0.00000743	1/20	0	1.00e-06	**0.00000584**	0.00000743	1/20	3420	8.70e-07	**0.00000584**	*0.00000606*	11/20	493	2.80e-07
delaunay_n16	65536	0.00244182	0.00254590	1/20	0	5.30e-05	0.00232641	0.00239002	1/20	0	4.30e-05	**0.00226376**	*0.00229371*	1/20	1292	2.44e-05
delaunay_n17	131072	0.00174222	0.00180321	1/20	0	3.30e-05	0.00161683	0.00166666	1/20	0	1.90e-05	**0.00156857**	*0.00161819*	1/20	2000	2.22e-05
delaunay_n18	262144	0.00122711	0.00128230	1/20	48	2.40e-05	0.00113724	0.00117144	1/20	0	1.80e-05	**0.00113332**	*0.00114381*	1/20	762	8.03e-06
delaunay_n19	524288	0.00087931	0.00090477	1/20	249	1.80e-05	0.00078757	0.00081717	1/20	47	1.17e-05	**0.00078693**	*0.00080017*	1/20	729	9.73e-06
delaunay_n20	1048576	0.00062085	0.00064504	1/20	65	1.30e-05	**0.00055890**	0.00057389	1/20	997	9.59e-06	0.00055891	*0.00056128*	9/20	27	3.50e-06
delaunay_n21	2097152	0.00043742	0.00045553	1/20	44	8.00e-06	**0.00039293**	0.00041812	1/20	1854	3.01e-05	**0.00039293**	*0.00039397*	8/20	45	1.35e-06
delaunay_n22	4194304	**0.00031352**	0.00032529	1/20	225	6.00e-06	**0.00031352**	0.00032530	1/20	2272	6.42e-06	**0.00031352**	*0.00031504*	13/20	286	2.61e-06
delaunay_n23	8388608	**0.00022443**	0.00023219	1/20	335	4.00e-06	**0.00022443**	0.00023219	1/20	2906	4.08e-06	**0.00022443**	*0.00022552*	8/20	526	1.30e-06
delaunay_n24	16777216	**0.00016180**	0.00016509	1/20	0	2.00e-06	**0.00016180**	0.00016509	1/20	3420	2.32e-06	**0.00016180**	*0.00016208*	11/20	1256	3.20e-07
co2010	201062	0.00067382	0.00072937	1/20	0	3.40e-05	0.00053700	0.00061060	1/20	0	5.00e-05	**0.00053442**	*0.00053453*	19/20	564	4.80e-07
la2010	204447	0.00036595	0.00042063	1/20	28	4.80e-05	**0.00017789**	0.00028104	3/20	35	7.40e-05	**0.00017789**	*0.00017789*	20/20	4	0
ia2010	216007	0.00067607	0.00071503	1/20	0	2.50e-05	0.00060102	0.00062450	1/20	0	1.90e-05	**0.00059147**	*0.00060046*	1/20	1239	4.62e-06
ks2010	238600	0.00057938	0.00063012	1/20	0	2.40e-05	**0.00051470**	0.00054080	1/20	0	1.60e-05	**0.00051470**	*0.00051875*	10/20	980	4.54e-06
tn2010	240116	0.00043051	0.00048415	1/20	0	3.00e-05	**0.00033671**	0.00038654	8/20	0	4.10e-05	**0.00033671**	*0.00033671*	20/20	4	0
az2010	241666	0.00057022	0.00062654	1/20	1	3.40e-05	0.00044358	0.00051398	1/20	130	4.10e-05	**0.00043839**	*0.00045183*	1/20	94	1.90e-05
al2010	252266	0.00057567	0.00065110	1/20	0	3.70e-05	0.00050982	0.00054469	1/20	69	2.70e-05	**0.00050732**	*0.00051154*	2/20	1295	4.01e-06
wi2010	253096	0.00053185	0.00062475	1/20	0	7.30e-05	0.00049099	0.00053505	1/20	106	4.70e-05	**0.00048247**	*0.00048581*	3/20	1374	4.78e-06
mn2010	259777	0.00061709	0.00066058	1/20	0	2.30e-05	0.00053535	0.00055552	1/20	161	1.30e-05	**0.00053491**	*0.00053926*	1/20	828	3.59e-06
in2010	267071	0.00054614	0.00060081	1/20	1	2.70e-05	**0.00045199**	0.00047766	7/20	51	3.20e-05	**0.00045199**	*0.00045199*	20/20	2	0
ok2010	269118	0.00054185	0.00059924	1/20	1	3.30e-05	0.00046031	0.00050597	1/20	192	2.10e-05	**0.00046030**	*0.00046677*	2/20	501	9.09e-06
va2010	285762	0.00060130	0.00066431	1/20	1	3.70e-05	**0.00050629**	0.00054526	6/20	129	3.60e-05	**0.00050629**	*0.00050810*	17/20	3	6.23e-06
nc2010	288987	0.00033045	0.00035826	1/20	1	2.10e-05	0.00029129	0.00030381	1/20	79	9.00e-06	**0.00029079**	*0.00029215*	1/20	1218	2.44e-06
ga2010	291086	0.00041692	0.00047249	1/20	74	4.10e-05	0.00036488	0.00039204	1/20	445	2.30e-05	**0.00035935**	*0.00036353*	1/20	1083	2.93e-06
mi2010	329885	0.00050347	0.00054874	1/20	1	2.50e-05	**0.00009247**	0.00020506	13/20	72	1.71e-04	**0.00009247**	*0.00009247*	20/20	1	0

Table 4: Continued

Instance		SaBTS [28]					MQI+SaBTS [28]					**MAMC**				
Graph	$\lvert V\rvert$	Φ_{best}	Φ_{avg}	hit	$t(s)$	σ	Φ_{best}	Φ_{avg}	hit	$t(s)$	σ	Φ_{best}	Φ_{avg}	hit	$t(s)$	σ
mo2010	343565	0.00054932	0.00061939	1/20	1	4.60e-05	**0.00044010**	0.00050980	1/20	82	4.20e-05	**0.00044010**	*0.00044334*	13/20	481	1.25e-05
oh2010	365344	0.00055701	0.00059543	1/20	2	2.20e-05	0.00047052	0.00050873	1/20	116	2.20e-05	**0.00045030**	*0.00047809*	1/20	312	9.88e-06
pa2010	421545	0.00034309	0.00038692	1/20	1	3.80e-05	**0.00026616**	0.00027731	11/20	0	2.30e-05	**0.00026616**	*0.00026616*	20/20	2	0
il2010	451554	0.00029324	0.00031372	1/20	1	1.60e-05	0.00025204	0.00026736	1/20	5	9.00e-06	**0.00024997**	*0.00025260*	2/20	515	1.74e-06
tx2010	914231	0.00028326	0.00032269	1/20	5	2.00e-05	0.00024258	0.00025289	1/20	2692	9.51e-06	**0.00024221**	*0.00024274*	2/20	176	4.60e-07
t60k	60005	0.00082876	0.00098630	1/20	9	1.07e-04	**0.00069385**	0.00073344	17/20	6	9.40e-05	**0.00069385**	*0.00069385*	20/20	1	0
wing	62032	0.00708296	0.00731614	1/20	0	1.56e-04	0.00668219	0.00689348	1/20	0	1.36e-04	**0.00654906**	*0.00662078*	1/20	1953	6.69e-05
brack2	62631	**0.00185332**	0.00192248	1/20	1767	4.80e-05	**0.00185332**	*0.00185332*	20/20	7	0	**0.00185332**	*0.00185332*	20/20	10	0
finan512	74752	**0.00062040**	*0.00062040*	20/20	0	0	**0.00062040**	*0.00062040*	20/20	0	0	**0.00062040**	*0.00062040*	20/20	3	0
fe_tooth	78136	0.00854413	0.00920313	1/20	2008	3.23e-04	0.00857068	0.00893257	1/20	735	2.09e-04	**0.00843147**	*0.00846699*	1/20	2032	8.13e-05
fe_rotor	99617	0.00319580	0.00322448	1/20	2081	2.00e-05	0.00303132	0.00313967	1/20	0	2.70e-05	**0.00302488**	*0.00303411*	14/20	542	2.59e-05
598a	110971	0.00323883	0.00327078	1/20	1707	1.90e-05	0.00325666	0.00326880	1/20	151	9.00e-06	**0.00323209**	*0.00323816*	2/20	2145	5.02e-06
fe_ocean	143437	0.00079436	0.00095124	1/20	2069	1.13e-04	**0.00078223**	0.00083713	7/20	1255	7.90e-05	**0.00078223**	*0.00078223*	20/20	47	0
144	144649	0.00611787	0.00620977	1/20	1444	5.20e-05	0.00610660	0.00620893	1/20	206	6.00e-05	**0.00605305**	*0.00607374*	1/20	2690	9.00e-06
wave	156317	0.00828070	0.00838166	1/20	2315	7.00e-05	0.00828692	0.00838030	1/20	764	7.80e-05	**0.00819101**	*0.00823630*	1/20	2867	2.25e-05
m14b	214765	0.00230194	0.00231493	1/20	818	5.00e-06	0.00229901	0.00232534	1/20	416	1.90e-05	**0.00228466**	*0.00229867*	1/20	1865	5.47e-06
auto	448695	0.00314093	0.00318418	1/20	1196	3.50e-05	**0.00298571**	0.00301256	4/20	120	6.00e-05	**0.00298571**	*0.00298579*	17/20	476	2.00e-07
coAuthorsDBLP	299067	0.04994323	0.05156218	1/20	252	8.40e-04	**0.00411522**	0.00494913	2/20	16	5.22e-04	**0.00411522**	*0.00411522*	20/20	10	0
thermal2	1227087	0.00027448	0.00028060	1/20	1568	3.99e-06	**0.00025296**	0.00025687	1/20	864	6.42e-06	**0.00025296**	*0.00025339*	10/20	223	4.90e-07
G3_circuit	1585478	0.00040793	0.00044424	1/20	1283	1.61e-05	**0.00036328**	0.00038075	1/20	741	1.51e-05	**0.00036328**	*0.00036393*	8/20	335	1.01e-06
ca-coauthors-dblp	540486	0.04062111	0.04297001	1/20	810	1.51e-03	**0.00226757**	0.00244841	15/20	108	3.14e-04	**0.00226757**	*0.00226757*	20/20	5	0
inf-roadNet-PA	1087562	0.00011467	0.00012343	1/20	1409	6.25e-06	**0.00008221**	0.00008789	2/20	418	7.57e-06	**0.00008221**	*0.00008230*	15/20	4	1.60e-07
sc-nasasrb	54870	0.00331924	0.00344521	1/20	2244	8.13e-05	**0.00325489**	0.00334089	2/20	2728	5.05e-05	**0.00325489**	*0.00325489*	20/20	14	0
sc-pkustk13	94893	0.00916596	0.00952165	1/20	2182	2.56e-04	0.00916933	0.00946360	1/20	1637	2.43e-04	**0.00905805**	*0.00906634*	15/20	1729	2.86e-05
soc-gowalla	196591	0.05394554	0.05548406	1/20	2160	8.51e-04	**0.01234567**	0.01247394	16/20	3286	2.57e-04	**0.01234567**	*0.01234567*	20/20	0	0
soc-twitter-follows	404719	0.07447639	0.07661000	1/20	2986	1.01e-03	**0.00540540**	0.01515060	1/20	1260	6.21e-03	**0.00540540**	*0.00540540*	20/20	38	0
soc-youtube	495957	0.05890796	0.05991584	1/20	1915	1.06e-03	**0.00892857**	0.01134011	12/20	550	3.25e-03	**0.00892857**	*0.00892857*	20/20	2	0
soc-flickr	513969	0.06114300	0.09731561	1/20	3319	2.96e-02	**0.00444444**	0.00565198	13/20	658	1.66e-03	**0.00444444**	*0.00444444*	20/20	1	0
soc-FourSquare	639014	0.32355245	0.32508644	1/20	1508	7.14e-04	**0.26530612**	0.26564625	15/20	743	5.89e-04	**0.26530612**	*0.26530612*	20/20	4	0
web-arabic-2005	163598	0.00001773	0.00003808	1/20	2555	3.60e-05	0.00000406	0.00000408	16/20	3189	6.00e-08	**0.00000306**	*0.00000306*	20/20	1	0

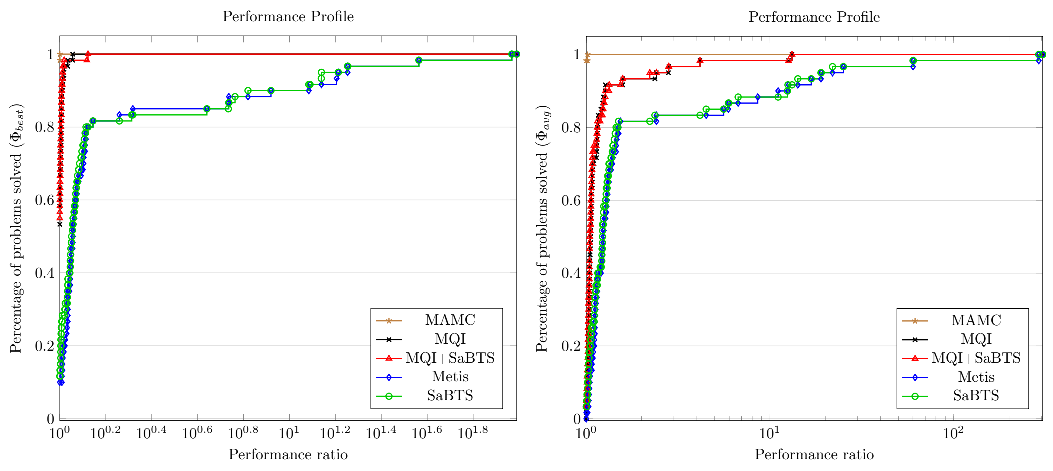

Fig. 3. Performance profiles of the proposed MAMC algorithm and the four reference algorithms Metis [20], MQI [21], SaBTS [28] and MQI+SaBTS [28] on the set of 60 benchmark instances. The left part corresponds to the best conductance Φ_{best} values, and the right part is for the average conductance Φ_{avg} values.

Table 5
Summary of comparative results between the proposed MAMC algorithm and each of the four reference algorithms Metis [20], MQI [21], SaBTS [28] and MQI+SaBTS [28].

Algorithm pair	#Instance	Indicator	#Wins	#Ties	#Losses	*p-value*
MAMC vs. Metis [20]	60	Φ_{best}	54	6	0	1.63e-10
		Φ_{avg}	60	0	0	1.63e-11
MAMC vs. MQI [21]	60	Φ_{best}	27	33	0	5.60e-06
		Φ_{avg}	60	0	0	1.63e-11
MAMC vs. SaBTS [28]	60	Φ_{best}	53	7	0	2.39e-10
		Φ_{avg}	58	1	1	2.93e-10
MAMC vs. MQI+SaBTS [28]	60	Φ_{best}	27	32	1	4.46e-06
		Φ_{avg}	58	2	0	3.51e-11

4.4 Application to complex network analysis

As indicated in the literature (e.g., [6,8,25,27]), conductance minimization is an useful tool for complex network analysis. As an additional benchmark assessment, we illustrate in this section the interest of the MAMC algorithm for the problem of community identification in complex networks. For this purpose, we consider 3 popular social networks: two with known (ground truth) community structures (Zachary's Karate Club and College Football Network) and one with unknown community structure (Bottlenose Dolphin Social Network).

Zachary's Karate Club [45]. This social network (Fig. 4) is from the well-

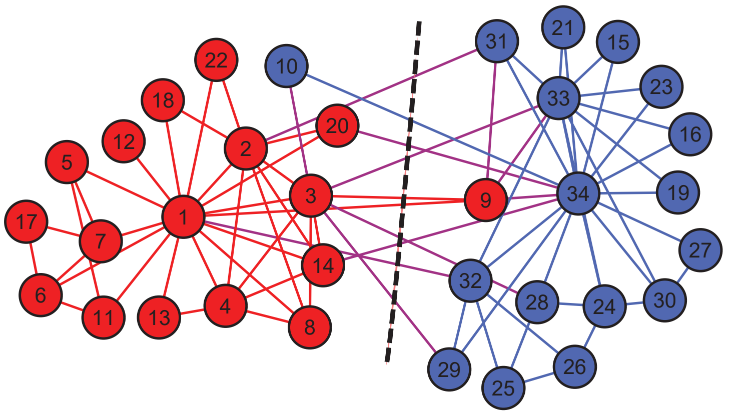

Fig. 4. Karate Club friendship network and the communities detected by the MAMC algorithm.

known karate club studied by Zachary, which represents the 34 members (nodes) and the 78 pairwise friendship relations (edges) between members observed over a period of three years. Due to a conflict between the administrator (node 34) and the instructor (node 1), the club members are divided into two groups (communities), each with half of the members. Fig. 4 shows the members (nodes) of the two groups in red and blue respectively. Running our MAMC algorithm on this network led to the partition indicated by the dash line with a conductance value of $\Phi(s) = 0.12820512$. As one can observe, the identified communities almost perfectly reflect the two groups observed by Zachary, with only 2 nodes (9 and 10) "incorrectly" assigned to the opposing group. This is reasonable and can be explained by the observations from Zachary. In fact, individual 9 is a weak political supporter of the club administrator before the fission, and individual 10 supports neither the administrator nor the instructor. As a result, none of individual 9 or 10 is solidly a member of either group. Finally, the partition identified by MAMC has the minimal conductance value ($\Phi(s) = 0.12820512$), while Zachary's partition has a larger conductance of $\Phi(s) = 0.14666666$.

College Football Network [14]. This social network (Fig. 5) is more complex and represents the schedule of American football games between Division IA colleges during the regular-season in Fall 2000. The network is composed of 115 nodes representing the teams and 613 links representing regular-season games between the two connected teams. The known communities are defined by the conferences with their composing teams and marked with different colors. In principle, teams from one conference are more likely to play games with each other than with teams from other conferences. There also exist some independent teams that do not belong to any conference, and these teams are marked with a dark blue color. The communities identified by the MAMC algorithm are represented by clusterings in Fig. 5. In general, MAMC correctly clusters teams of each conference. The independent teams are clustered

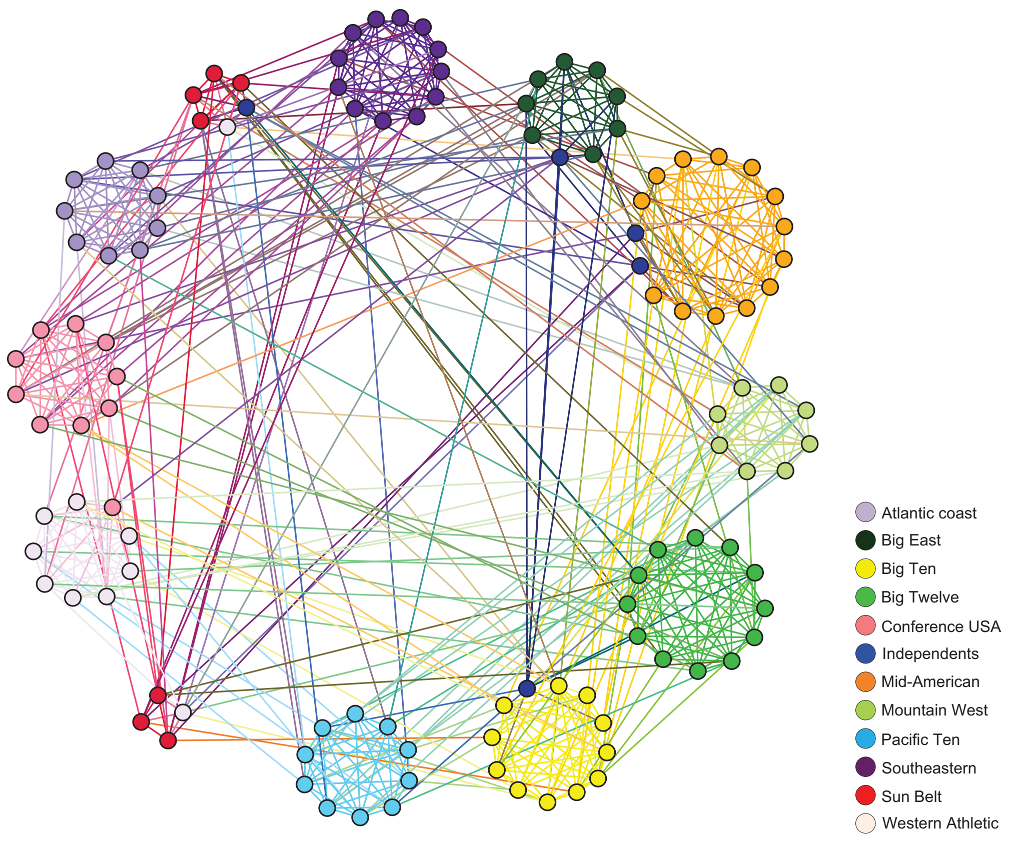

Fig. 5. College Football Network. The colors indicate different conferences, and clusterings show the communities identified by the MAMC algorithm.

with conferences with which they played games most frequently, because the independent teams seldom play games between themselves. The clusters detected by MAMC deviate slightly from the conference partition in several ways. First, the Sun Belt conference, marked in red, is split into two parts, which are grouped with Western Athletic and Independents conference due to the fact that there was only one game involving teams from these two parts. Second, one team from Conference USA (colored in pink) is clustered with teams from the Western Athletic conference. This team played no games with other teams from the Conference USA, but played games with every team from the Western Athletic conference. Third, two teams from the Western Athletic conference are isolated from other teams from this conference. The team at the upper position had no intra-conference game, and the team at the lower position had only 2 intra-conference games, but both teams had inter-conference games with every member of the cluster they are assigned to. In summary, our MAMC algorithm perfectly identified the community structures established in the regular-season-game association, and in addition, detected the lack of intra-conference association that the known community structure fails to represent.

Bottlenose Dolphin Social Network [29]. This social network (Fig. 6) is

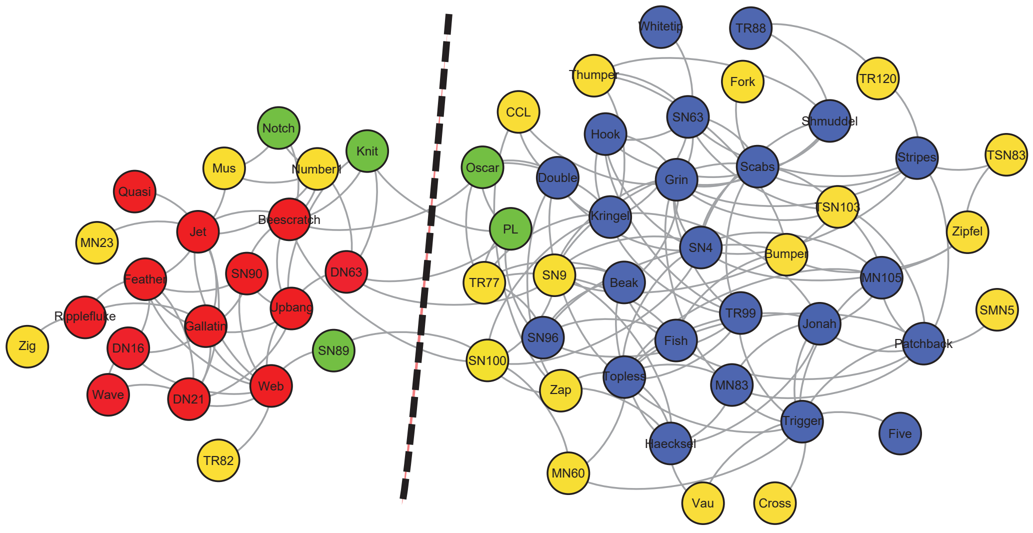

Fig. 6. Bottlenose Dolphin Social Network. The nodes are colored based on the groups observed in the study by Lusseau et al. [29]. The clusterings represent communities detected by the MAMC algorithm.

composed of 62 bottlenose dolphins living off Doubtful Sound, New Zealand. The 159 social associations between dolphin pairs are established based on direct observations conducted during a period of seven years by Lusseau et al. [29]. In this network, nodes represent dolphins and links represent social associations. The 3 groups of 40 dolphins observed by Lusseau et al. tend to spend more time together than with others, and are colored in green, red and blue respectively, while the dolphins with yellow color are not involved in the clustering analysis. The dash line denotes the community division found by our MAMC algorithm. We can see that the achieved division matches well the observed groups, separating the red and blue groups into two communities. The green group is split evenly between the two detected communities, because this group is a weak group and is not well represented by the social network since most of its members share no social associations [29].

5 Analysis

In this section, we first analyze the key parameters of the proposed algorithm, and the impacts of the progressive constrained neighborhood, the pool initialization procedure, and the pool updating procedure.

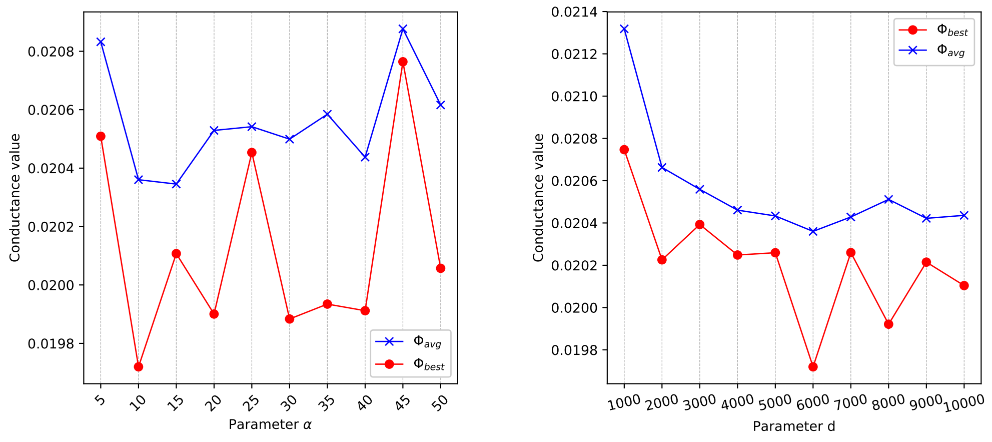

Fig. 7. Average values of Φ_{best} and Φ_{avg} on six hard instances obtained by executing MAMC with different values of parameters α and d. The left part concerns the parameter α with values $\alpha \in \{5, 10, \ldots, 45, 50\}$, and the right part concerns the parameter d with values $d \in \{1000, 2000, \ldots, 9000, 10000\}$.

5.1 Study of the parameters for tabu search

The progressive constrained neighborhood tabu search (PCNTS) of MAMC requires two parameters (tabu tenure management factor α and depth of tabu search d). To study the effect of these parameters and determine a proper value, we performed the following experiment. For each parameter, we varied its values within a reasonable range, while maintaining the other parameter to their default values as shown in Table 2. Specifically, α varied its values in $\{5, 10, \ldots, 45, 50\}$, while d takes its values in $\{1000, 2000, \ldots, 9000, 10000\}$. The experiment in this section was carried out on six hard instances with different characteristics (*smallworld*, *delaunay_n18*, *oh2010*, *wave*, *sc-pkustk13*, *web-arabic-2005*). We ran our algorithm with each value of these parameters 20 times to solve each instance with a cutoff time of 60 minutes. Fig. 7 shows the average values of Φ_{best} and Φ_{avg} over these six instances, where the X-axis indicates the parameter values and the Y-axis shows the best and average conductance values.

From Fig. 7, we observe that the performance of MAMC is significantly influenced by the setting of each parameter. For α, the best performance is attained when $\alpha = 10$, and a too small or too large α value leads to a poor performance of MAMC. This can be explained by the fact that a small (or large) α value makes the prohibited time too short (or too long). For d, the value of 6000 is the best choice, and a too large or too small d value deteriorates MAMC performance. Thus, we set $\alpha = 10$ and $D = 6000$ in our experiment.

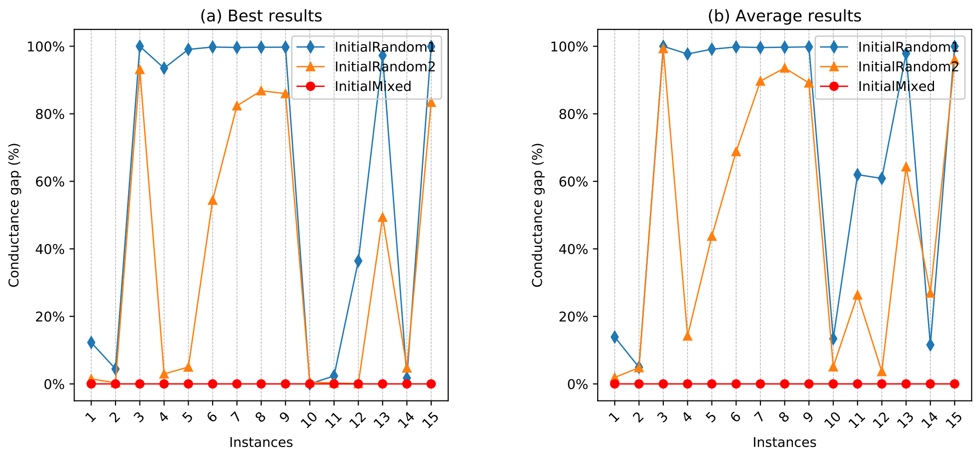

Fig. 8. Comparison of the quality-and-diversity based population initialization (InitialMixed) with two random initialization variants (InitialRandom1 and InitialRandom2).

5.2 Quality-and-diversity based initialization v.s. random initialization

The MAMC algorithm uses a mixed population initialization strategy that takes into account both quality and diversity (see Section 3.2). We study the effect of this strategy by comparing it with two random initializations. The first one is a pure random initialization (denoted by InitialRandom1), while the second one uses the progressive constrained neighborhood tabu search of Section 3.4 to improve each random seeding solution (denoted by InitialRandom2). We tested these two variants within our MAMC algorithm based on 15 instances arbitrarily chosen from the benchmark set and compared the results with the quality-and-diversity based initialization (denoted by InitialMixed). The plots of their best and average conductance gaps are shown in Fig. 8, where the X-axis indicates the instances (named by numbers from 1 to 15) and the Y-axis shows the best (or average) conductance gap in percentage. The best (or average) conductance gap is calculated as $(\Phi_{InitialRandomK} - \Phi_{MAMC})/\Phi_{InitialRandomK} \times 100\%$ $(K = 1, 2)$ where $\Phi_{InitialRandomK}$ and Φ_{MAMC} are the best (or average) conductance values of the variant $(InitialRandomK)$ and MAMC respectively. From Fig. 8, we observe that MAMC with the quality-and-diversity initialization outperforms the two random initializations. Moreover, the pure random initialization led to the worst results. This experiment confirms the usefulness of our designed mixed initialization procedure.

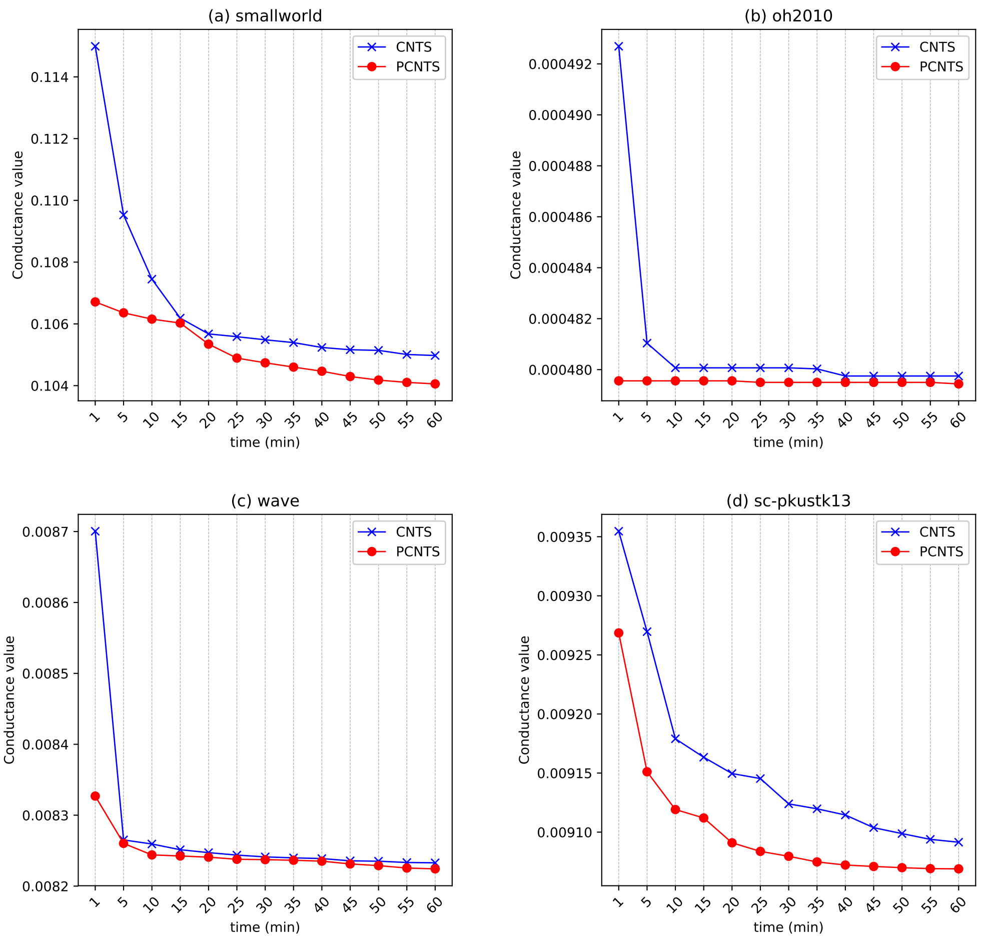

Fig. 9. Comparison of MAMC with the progressive constrained neighborhood (denoted by PCNTS) and its variant with the whole neighborhood (denoted by CNTS) according to their running profiles (convergence graphs).

5.3 Effectiveness of the progressive constrained neighborhood

As described in Section 3.4, the MAMC algorithm employs a progressive constrained neighborhood in its PCNTS procedure. To assess the usefulness of this progressive constrained neighborhood, we created a MAMC variant (denoted by CNTS) where the tabu search procedure of Algorithm 3 examines at each iteration the whole constrained neighborhood induced by the whole set of critical vertices $CV(s)$ (i.e., by setting $ev = |CV(s)|$ in Algorithm 3). To compare MAMC and this variant, we ran both algorithm on four selected instances (*smallworld*, *oh2010*, *wave*, *sc-pkustk13*) and show their running profiles (convergence graphs) in Fig. 9.

We observe that thanks to the progressive constrained neighborhood, the MAMC algorithm has a better convergence throughout the search from the

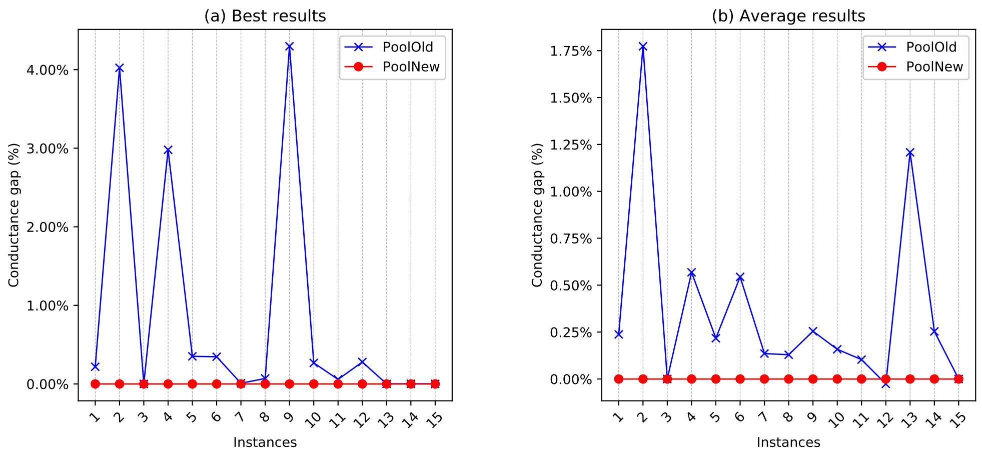

Fig. 10. Comparison of the distance-and-quality based pool updating procedure (PoolNew) and its variant only based on solution quality (PoolOld).

beginning to the end of the given time budget. This holds for all four instances. This experiment confirms the relevance of the adopted progressive constrained neighborhood within the tabu search based local optimization procedure.

5.4 Effectiveness of the distance-and-quality based pool updating procedure

MAMC uses a distance-and-quality based pool updating procedure (see Section 3.5) to maintain a healthy population. We present an experiment to assess the effectiveness of this pool updating strategy (denoted by PoolNew) by making a comparison with the traditional quality based pool updating strategy (denoted by PoolOld) which replaces the worst solution in the population with the offspring solution.

We tested both strategies within our MAMC algorithm based on 15 instances randomly taken from the benchmark set. Fig. 10 provides the plots of their best and average conductance gap. The X-axis indicates the name of instances (named by numbers from 1 to 15). The Y-axis shows the best (or average) conductance gap in percentage, calculated as $(\Phi_{PoolOld} - \Phi_{MAMC})/\Phi_{PoolOld} \times 100\%$ where $\Phi_{PoolOld}$ is the best (or average) conductance value of the algorithm variant (PoolOld) and Φ_{MAMC} is the best (or average) conductance value of MAMC.

From Fig. 10, we observe that the MAMC algorithm performs generally better with the distance-and-quality based pool updating procedure than with the traditional quality pool updating procedure. This outcome confirms that the distance-and-quality strategy plays a positive role and contributes to the performance of the MAMC algorithm.

6 Conclusion and prospective

The minimum conductance partitioning problem has a number of significant applications in various domains. In this work, we introduced an effective hybrid evolutionary algorithm for finding low conductance for large graphs. Based on the population memetic search framework, the proposed algorithm integrates some important features. First, to ensure an effective and efficient intensification of its local optimization component on large and massive graphs, the algorithm adopts an original progressive neighborhood whose size is dynamically adjusted according to the search state. Second, to maintain a healthy population of high diversity and good quality, the algorithm uses a proven distance-and-quality pool updating strategy for its population management. Finally, to further diversify the search, a conventional crossover is applied to generate offspring solutions.

We showed that the proposed algorithm competes very favorably with the current best-performing algorithms when it is assessed on 60 large-scale real-world benchmark instances (including 50 graphs from the 10th DIMACS Implementation Challenge and 10 graphs from the Network Data Repository online, with up to 23 million vertices). The computational results led to the main conclusions that 1) our algorithm dominates the reference algorithms for the tested instances, and 2) it is able to further improve the solutions obtained by the popular max-flow quotient-cut improvement algorithm. As an application example, we showed the proposed algorithm can be used to detect meaningful community structures in complex networks. Finally, we investigated the essential components of the algorithm, leading to the findings that 1) the progressive neighborhood used by the local optimization procedure plays an important role of ensuring a fast convergence of the algorithm, and 2) the quality-and-diversity pool initialization and distance-and-quality pool management help the algorithm to maintain a healthy population, which contributes to its performance.

For future work, several potential research lines can be followed. First, the idea of the progressive neighborhood strategy is of general interest, it would be interesting to test the idea in other settings, in particular related to large graph optimization. Second, it would also be worth studying the combination of max-flow methods like MQI and multilevel optimization. Third, few studies exist on exact approaches for MC-GPP. Research in this area is clearly needed.

Acknowledgment

We are grateful to the reviewers for their useful comments and suggestions which helped us to significantly improve the paper. Support from the China Scholarship Council (CSC) for the first author is also acknowledged.

References

[1] Andersen, R., & Lang, K. J. An algorithm for improving graph partitions. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 651-660, 2008.

[2] Arora, S., Rao, S., & Vazirani, U. $O(\sqrt{logn})$ approximation to sparsest cut in õ(n^2) time. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 238-247, 2004.

[3] Arora, S., Rao, S., & Vazirani, U. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2), 5:1-5:37, 2009.

[4] Benlic, U., & Hao, J. K. Una Benlic and Jin-Kao Hao. A multilevel memetic approach for improving graph k-partitions. *IEEE Transactions on Evolutionary Computation*, 15(5): 624-642, 2011.

[5] Benlic, U., & Hao, J. K. Memetic search for the quadratic assignment problem. *Expert Systems with Applications*, 42(1): 584-595, 2015.

[6] Benson, A. R., Gleich, D. F., & Leskovec, J. Higher-order organization of complex networks. *Science*, 353(6295), 163-166, 2016.

[7] Chalupa, D. A memetic algorithm for the minimum conductance graph partitioning problem. *arXiv preprint* arXiv: 1704.02854, 2017.

[8] Chalupa, D., Hawick, K. A., & Walker, J. A. Hybrid bridge-based memetic algorithms for finding bottlenecks in complex networks. *Big Data Research*, 14, 68-80, 2018.

[9] Cheeger J. A lower bound for the smallest eigenvalue of the Laplacian. In *Proceedings of the Princeton Conference in Honor of Professor S. Bochner*, pages 195-199, 1969.

[10] Cheng, D., Kannan, R., Vempala, S., & Wang, G. A Divide-and-Merge Methodology for Clustering. *ACM Transactions on Database Systems*, 31(44), 1499-1525, 2006.

[11] Dolan, E. D., & Moré, J. J. enchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2), 201-213, 2002.

[12] Fortunato, S. Community detection in graphs. *Physics Reports*, 486(3-5), 75-174, 2010.

[13] Galinier, P., Boujbel, Z., & Fernandes, M. C. An efficient memetic algorithm for the graph partitioning problem. *Annals of Operations Research*, 191(1), 1-22, 2011.

[14] Girvan, M., & Newman, M. E. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12), 7821-7826, 2002.

[15] Guerrero, C., Lera, I., & Juiz, C. Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures. *Future Generation Computer Systems*, 97, 131-144, 2019.

[16] Gusfield, D. Partition-distance: A problem and class of perfect graphs arising in clustering. *Information Processing Letters*, 82(3), 159-164, 2002.

[17] Hao, J. K. Memetic Algorithms in Discrete Optimization. In Neri, F., Cotta, C., &Moscato P. (Eds.) Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Chapter 6, pages 73-94, 2012.

[18] Hochbaum, D. S. Polynomial time algorithms for ratio regions and a variant of normalized cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5), 889-898, 2010.

[19] Hochbaum, D. S. A polynomial time algorithm for rayleigh ratio on discrete variables: Replacing spectral techniques for expander ratio, normalized cut, and cheeger constant. *Operations Research*, 61(1), 184-198, 2013.

[20] Karypis G., & Kumar V. MeTiS 5.1.0: Unstructured graphs partitioning and sparse matrix ordering system. *Technical Report*, Department of Computer Science, University of Minnesota, 1998.

[21] Lang, K., & Rao, S. A flow-based method for improving the expansion or conductance of graph cuts. In *International Conference on Integer Programming and Combinatorial Optimization*, 325-337, Springer, Berlin, Heidelberg, 2004.

[22] Leighton, T., & Rao, S. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6), 787-832, 1999.

[23] Leskovec, J., Lang, K. J., Dasgupta, A., & Mahoney, M. W. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International Conference on World Wide Web*, pages 695-704, ACM, 2008.

[24] Leskovec, J., Lang, K. J., Dasgupta, A., & Mahoney, M. W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1), 29-123, 2009.

[25] Li, P., & Milenkovic, O. (2017). Inhomogeneous hypergraph clustering with applications. In *Advances in Neural Information Processing Systems*, pp. 2308-2318, 2017.

[26] Lü, Z., Glover, F., & Hao, J. K. A hybrid metaheuristic approach to solving the UBQP problem. *European Journal of Operational Research*, 207(3), 1254-1262, 2002.

[27] Lu, Z., Wahlström, J., & Nehorai, A. Community detection in complex networks via clique conductance. *Scientific Reports*, 8(1), 5982, 2018.

[28] Lu, Z., Hao, J. K., & Zhou, Y. Stagnation-aware breakout tabu search for the minimum conductance graph partitioning problem. *Computers & Operations Research*, 111, 43-57, 2019.

[29] Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., & Dawson, S. M. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4), 396-405, 2003.

[30] Moalic, L., & Gondran, A. Variations on memetic algorithms for graph coloring problems. *Journal of Heuristics*, 24 (1), 124, 2018.

[31] Moura, D. L. L., Cabral, R. S., Sales, T., & Aquino, A. L. L. An evolutionary algorithm for roadside unit deployment with betweenness centrality preprocessing. *Future Generation Computer Systems*, 88, 776-784, 2018.

[32] Neri, F., Cotta, C., & Moscato, P. (Eds.). Handbook of memetic algorithms (Vol. 379). Berlin: Springer, 2012.

[33] Porumbel, D. C., Hao, J. K., Kuntz, P. An Evolutionary Approach with Diversity Guarantee and Well-Informed Grouping Recombination for Graph Coloring. *Computers & Operations Research*, 37(10), 1822-1832, 2010.

[34] Rossi, R., & Ahmed, N. The network data repository with interactive graph analytics and visualization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 4292-4293, 2015.

[35] Schaeffer, S. E. Graph clustering. *Computer Science Review*, 1(1), 27-64, 2007

[36] Shi, J., & Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888-905, 2000.

[37] Šíma, J., & Schaeffer, S. E. On the NP-completeness of some graph cluster measures. In *Proceedings of the 32nd Conference on Current Trends in Theory and Practice of Computer Science*, Lecture Notes in Computer Science book series, volume 3831, pages 530-537, Springer, Berlin, Heidelberg, 2006.

[38] Siqueira, A. S., da Silva, R. C., & Santos, L. R. Perprof-py: A python package for performance profile of mathematical optimization software. *Journal of Open Research Software*, 4(1), 2016.

[39] Spielman, D. A., & Teng, S. H. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1), 1-26, 2013.

[40] Teylo, L., de Paula, U., Frota, Y., de Oliveira, D., & Drummond, L. M. A. A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds. *Future Generation Computer Systems*, 76, 1-17, 2017.

[41] Van Laarhoven, V. T., & Marchiori, E. Local network community detection with continuous optimization of conductance and weighted kernel k-means. *Journal of Machine Learning Research*, 317(1), 5148-5175, 2016.

[42] Voevodski, K., Teng, S. H., & Xia, Y. Finding local communities in protein networks. *BMC Bioinformatics*, 10(1), 297, 2009.

[43] Wu, Q., & Hao, J. K. Memetic search for the max-bisection problem. *Computers & Operations Research*, 40(1), 166-179, 2013.

[44] Yang, J., & Leskovec, J. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1), 181-213, 2015.

[45] Zachary, W. W. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4), 452-473, 1977.

[46] Zhu, Z. A., Lattanzi, S., & Mirrokni, V. S. A local algorithm for finding well-connected clusters. In *Proceedings of the 30th International Conference on Machine Learning*, pages 396-404, 2013.