Iterated Multilevel Simulated Annealing for Large-Scale Graph Conductance Minimization

Zhi Lu [a,b], Jin-Kao Hao [b,*] Una Benlic [c], David Lesaint [b]

[a] *Business School, University of Shanghai for Science & Technology, 516 Jungong Rd., Shanghai 200093, China*

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

[c] *Tesco PLC, Lever Building, 85 Clerkenwell Rd., Holborn, London EC1R 5AR, UK*

Accepted to *Information Sciences, April 2021*

Abstract

Given an undirected connected graph $G = (V, E)$ with vertex set V and edge set E, the minimum conductance graph partitioning problem is to partition V into two disjoint subsets such that the conductance, i.e., the ratio of the number of cut edges to the smallest volume of two partition subsets is minimized. This problem has a number of practical applications in various areas such as community detection, bioinformatics, and computer vision. However, the problem is computationally challenging, especially for large problem instances. This work presents the first iterated multilevel simulated annealing algorithm for large-scale graph conductance minimization. The algorithm features a novel solution-guided coarsening method and an effective solution refinement procedure based on simulated annealing. Computational experiments demonstrate the high performance of the algorithm on 66 very large real-world sparse graphs with up to 23 million vertices. Additional experiments are presented to get insights into the influences of its algorithmic components. The source code of the proposed algorithm is publicly available, which can be used to solve various real world problems.

Keywords: Combinatorial optimization; large-scale optimization; metaheuristics; heuristics; graph theory.

* Corresponding author. Email: jin-kao.hao@univ-angers.fr

1 Introduction

Graph partitioning problems are popular and general models that are frequently used to formulate numerous practical applications in various domains. Given an undirected graph with a vertex set and an edge set, graph partitioning is to divide the vertex set into two or more disjoint subsets while meeting a defined objective. For example, the popular NP-hard (non-deterministic polynomial-time hardness) 2-way graph partitioning problem is to minimize the number of edges crossing the two partition subsets [12].

The minimum conductance graph partitioning problem (MC-GPP) studied in this work is another typical graph partitioning problem stated as follows. Let $G = (V, E)$ be an undirected connected graph with vertex set V and edge set E. A cut in G is a partition of its vertex set V into two disjoint subsets S and $\overline{S} = V \setminus S$, while the cut-set is the set of edges that have one endpoint in each subset of the partition. By convention, a cut is denoted by $s = (S, \overline{S})$ and its cut-set is denoted by $cut(s)$.

Given a cut $s = (S, \overline{S})$ from the search space Ω composed of all the possible cuts of a graph G, its *conductance* $\Phi(s)$ is the ratio between the number of cut edges and the smallest volume of the two partition subsets, i.e.,

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

where $vol(S) = \sum\limits_{v \in S} deg(v)$ and $vol(\overline{S}) = \sum\limits_{v \in \overline{S}} deg(v)$ are the volume of S and $\overline{S}$, respectively, and $deg(v)$ is the number of vertices incident to v in G. The conductance of a graph G is the minimum conductance over all the cuts of the graph: $\Phi(G) = min_{s \in \Omega} \Phi(s)$. In mathematics and statistical physics, the conductance is also called the Cheeger constant, Cheeger number or isoperimetric number [9] with a different denominator that depends upon the number of vertices in S and $\overline{S}$.

The *minimum conductance graph partitioning problem* (MC-GPP) is then to determine the conductance of an arbitrary graph G, i.e., find the cut s^* in G such that $\Phi(s^*) \leq \Phi(s)$ for any s in Ω.

$$\text{(MC-GPP)} \quad s^* = \arg\ \min_{s \in \Omega} \Phi(s) \tag{2}$$

In terms of the computational complexity theory, MC-GPP is known to be an NP-hard problem [34], and thus computationally challenging. From a practical perspective, MC-GPP has a number of relevant applications in different fields such as clustering [10], community detection [11], bioinformatics [39], computer vision [33], and large graph size estimation [28]. As the result, designing effective solution methods for MC-GPP is both challenging and important.

Given the relevance of MC-GPP, considerable efforts have been dedicated to developing various algorithms for solving the problem, which mainly fall into three families: exact, approximation, and heuristic algorithms. Time-efficient exact algorithms were introduced in [18] to solve a variant of the conductance problem and other ratio region problems in the context of image segmentation. Later, a polynomial time algorithm for the Rayleigh ratio minimization problem on discrete variables was presented in [19]. The first (and considered weak) approximation algorithm for MC-GPP was presented in [9]. Improved approximation algorithms were studied including an $\mathcal{O}(log(|V|))$-approximation algorithm [25], and an $\mathcal{O}(\sqrt{log(|V|)})$-approximation algorithm [3].

Several algorithms with performance guarantees for conductance minimization were also proposed for large-scale graph clustering in social and information networks. For instance, local algorithms were presented in [35] for clustering massive graphs based on the conductance criterion. Another local algorithm was introduced in [42] that is able to find well-connected clusters for large graphs in terms of the clustering accuracy and the conductance of the output set. The conductance measure was used to characterize the "best" possible community in [26] where approximation algorithms with performance guarantee were proposed for the related graph partitioning problem over a wide range of size scales. Although these algorithms can theoretically provide provable performance guarantees on the quality of the obtained solutions, they are either designed for specific cases or require large computation time for large graphs due to the intrinsic intractability of MC-GPP.

To handle large problems that cannot be solved by exact methods, heuristic methods provide an interesting alternative approach to produce high-quality (not necessarily optimal) solutions within a reasonable amount of computation time. In particular, a number of generic metaheuristics (see e.g., [7]) are available including single-solution based methods (e.g., simulated annealing and tabu search) and population-based methods (e.g., evolutionary algorithms and particle swarm optimization). These methods have been used to tackle various large and computationally complex problems [13]. However, the success of these methods depends strongly on a careful design and dedicated adaptations of the methods to the problem at hand [6]. For MC-GPP, several heuristic and metaheuristic algorithms have been proposed, which are reviewed as follows. A max-flow quotient-cut improvement algorithm (MQI) was introduced in [24] to refine an initial cut of the Metis graph partitioning heuristic [22]. This work was extended in [2], which solves a sequence of minimum cut problems to find a larger-than-expected intersection with lower conductance. In [27], a minus top-k partition (MTP) method was studied for discovering a global balanced partition with low conductance. In [38], a continuous optimization approach was proposed for conductance minimization in the context of local network community detection. The first metaheuristic algorithms for MC-GPP including simple local search and basic memetic algorithms were studied in [8], which were used for finding bottlenecks in complex networks. A stagnation-aware breakout tabu search algorithm (SaBTS) was presented in [30], which combines a dedicated tabu search procedure to discover high-quality solutions and a self-adaptive perturbation procedure to overcome hard-to-escape local optimum traps. Computational results were reported on real world graphs with up to 500,000 vertices. Recently, a hybrid evolutionary algorithm (MAMC) with a powerful local search and a quality-and-diversity based pool management was introduced in [29], which reported new results on large graphs with up to 23 million vertices. These studies greatly contributed to better solving the given problem. However, two main limitations can be identified in the existing approaches, which motivate the current work. First, these approaches have difficulties in robustly and consistently producing high-quality solutions for large-scale graphs with more than several million vertices. Second, they may require a substantial amount of computation time to reach satisfactory solutions.

On the other hand, the multilevel approach [40] is known to be a powerful framework for large graph partitioning in various domains (e.g., complex network analysis [37]). However, this approach has not been investigated for solving MC-GPP. This work aims to fill in the gap by presenting an effective multilevel algorithm for MC-GPP that is able to produce high-quality solutions within a reasonable time frame for large-scale sparse graphs arising from real world applications. The contributions can be summarized as follows.

First, an iterated multilevel simulated annealing algorithm (IMSA) is introduced for MC-GPP. The algorithm consists of a novel solution-guided coarsening method and a powerful local refinement procedure to effectively sample the search space of the problem.

Second, extensive computational assessments are presented on three sets of 66 very large real-world benchmark instances (including 56 graphs from the 10th DIMACS Implementation Challenge and 10 graphs from the Network Data Repository online, with up to 23 million vertices). The computational experiments indicate a high competitiveness of the proposed IMSA algorithm compared to the existing state-of-the-art algorithms.

Third, the code of the IMSA algorithm will be publicly available, which can help researchers and practitioners to better solve various practical problems that can be formulated as MC-GPP.

The remainder of the paper is structured as follows. Section 2 presents the

IMSA algorithm. Section 3 shows the computational studies and comparisons between the proposed IMSA algorithm and state-of-the-art algorithms. Section 4 provides analyses of the key algorithmic components. Section 5 presents concluding remarks and perspectives for future research.

2 Iterated multilevel simulated annealing

This section presents the proposed iterated multilevel simulated annealing algorithm for MC-GPP. After illustrating its general algorithmic framework, the key algorithmic components are described.

2.1 General framework

The multilevel approach is a general framework that has shown to be very successful in tackling the classic graph partitioning problem. Generally, this approach consists of three basic phases (coarsening, initial partitioning, and uncoarsening) [17]. During the coarsening phase, the given graph is successively reduced to obtain a series of coarsened graphs with a decreasing number of vertices. An initial partition is then generated for the coarsest graph. Finally, during the uncoarsening phase, the coarsened graphs are successively unfolded in the reverse order of the coarsening phase. For each uncoarsened graph, the partition of the underlying coarsened graph is projected back to the uncoarsened graph and then improved by a refinement procedure. This process stops when the input graph is recovered and its partition is refined. In [40], Walshaw introduced iterated multilevel partitioning (analogous to the use of V-cycles in multigrid methods [36]) where the coarsening-uncoarsening process is iterated and the partition of the current iteration is used for the whole coarsening phase of the next iteration.

The proposed IMSA algorithm for MC-GPP is based on the ideas described above and includes two original features: 1) a solution-guided coarsening method, and 2) a powerful refinement procedure which is applied during both the coarsening and the uncoarsening phase. Fig. 1 illustrates the iterated multilevel framework adopted by the proposed IMSA algorithm, while the entire algorithmic framework is presented in Algorithm 1. Informally, IMSA performs a series of V-cycles, where each V-cycle is composed of a coarsening phase and an uncoarsening phase, mixed with local refinement for each intermediate (coarsened and uncoarsened) graph.

It is worth noting that IMSA requires a seeding partition s_0 of the input graph to initiate its first iteration (the first V-cycle). Generally, the seeding

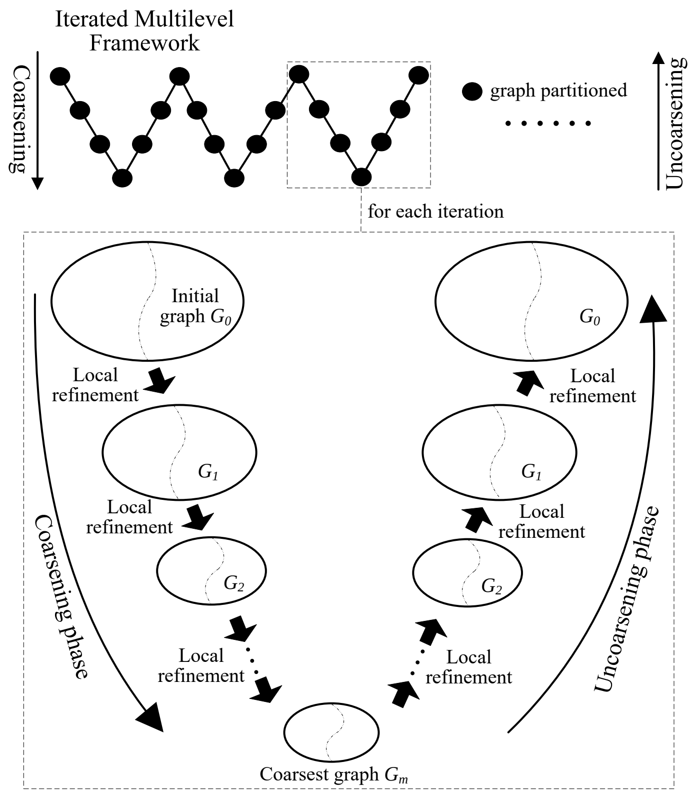


Fig. 1. An illustration of the iterated multilevel framework for the proposed algorithm.

partition can be provided by any means. For instance, for the experimental studies reported in Section 3, the (fast) MQI algorithm [24] was adopted. For each subsequent iteration, the best partition found during the previous V-cycle then serves as the new seeding partition. The whole process terminates when a given stopping condition (e.g., cutoff time limit, maximum number of V-cycles) is reached and the global best partition found during the search is returned.

2.2 Solution-guided coarsening procedure

Given a graph $G = (V, E)$ (renamed as $G_0 = (V_0, E_0)$) and the seeding partition s_0, the solution-guided coarsening phase progressively transforms G_0 into smaller intermediate graphs $G_i = (V_i, E_i)$ such that $|V_i| > |V_{i+1}|$ ($i = 0, \dots, m-1$), until the last coarsened graph G_m becomes sufficiently small (i.e., the number of vertices in V_m is below a given threshold ct). During the coarsening process, all intermediate graphs are recorded for the purpose of uncoarsening.

Generally, to generate a coarsened graph G_{i+1} from G_i, the coarsening process performs two consecutive steps: an edge matching step and an edge contraction

Algorithm 1 Iterated multilevel simulated annealing (IMSA) for MC-GPP.

Require: Graph $G = (V, E)$, seeding partition s_0, coarsening threshold ct.
Ensure: The best partition s^* found during the search.

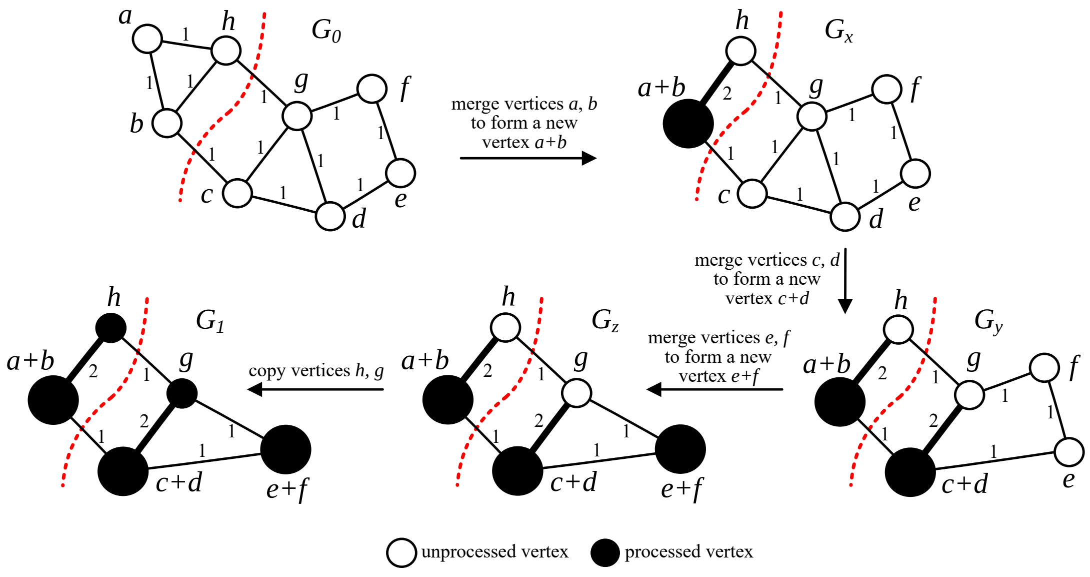
```
1: $s^* \leftarrow s_0$
2: $i \leftarrow 0$
3: repeat
4:     while $|V_i| > ct$ do
5:         $(G_{i+1}, s_{i+1}) \leftarrow Solution\_Guided\_Coarsening(G_i, s_i)$      /* Section 2.2 */
6:         $s_{i+1} \leftarrow Local\_Refinement(s_{i+1})$                   /* Sections 2.4 and 2.5 */
7:         $i \leftarrow i + 1$
8:     end while
9:     while $i > 0$ do
10:        $i \leftarrow i - 1$
11:        $(G_i, s_i) \leftarrow Uncoarsening(G_{i+1}, s_{i+1})$                      /* Section 2.3 */
12:        $s_i \leftarrow Local\_Refinement(s_i)$
13:    end while
14:    if $\Phi(s_i) < \Phi(s^*)$ then
15:        $s^* \leftarrow s_i$
16:    end if
17: until Stopping condition is met
18: return $s^*$
```

step. For each V-cycle, the initial graph $G_0 = (V_0, E_0)$ is supposed to have a "unit weight" for all the vertices and edges.

The edge matching step aims to find an independent set of edges $M \subset E_i$ such that the endpoints of any two edges in M are not adjacent. For this purpose, IMSA adopts the fast *Heavy-Edge Matching* (HEM) heuristic [21] with a time complexity of $\mathcal{O}(|E|)$. HEM considers vertices in a random order, and matches each unmatched vertex v with its unmatched neighbor vertex u such that 1) v and u are in the same subset of the current partition s_i; and 2) edge $\{v, u\}$ has the maximum weight over all edges incident to v (ties are broken randomly). In this work, edge matching is guided by the current partition s_i of G_i in the sense that the cut edges in the cut-set $cut(s_i)$ are ignored and only vertices of the same partition subset are considered for matching.

The edge contraction step collapses the endpoints of each edge $\{v_a, v_b\}$ in M to form a new vertex $v_a + v_b$ in the coarsened graph G_{i+1}, while vertices which are not endpoints of any edge of M are simply copied over to G_{i+1}. For the new vertex $v_a + v_b \in V_{i+1}$, its weight is set to be the sum of the weights of v_a and v_b. The edge between v_a and v_b is removed, and the edges incident to v_a and v_b are merged to form a new edge in E_{i+1} with a weight that is set to be the sum of the weights of the merged edges.

Once the coarsened graph G_{i+1} is created, the partition s_i of G_i is projected onto G_{i+1}, followed by improvement with the local refinement procedure. The



a
h
b
c
d
e
f
g
G_0
merge vertices a, b to form a new vertex a+b
a+b
G_x
merge vertices c, d to form a new vertex c+d
G_y
c+d
merge vertices e, f to form a new vertex e+f
G_z
e+f
copy vertices h, g
G_1
unprocessed vertex
processed vertex

Fig. 2. An example of the solution-guided coarsening process to create a coarsened graph.

improved partition is then used to create the next coarsened graph.

Fig. 2 illustrates how the first coarsened graph is created from an initial graph G_0 with 8 vertices (unit weight for both vertices and edges) and the partition $s_0 = (\{a, b, h\}, \{c, d, e, f, g\})$ (indicated by the red dashed line). The edge matching step first uses the HEM heuristic to identify the set of independent edges $M = (\{a, b\}, \{c, d\}, \{e, f\})$ guided by s_0. Then for each edge in M, say $\{a, b\}$, its endpoints are merged to form a new vertex $a + b$ in G_1 with vertex weight $w(a + b) = w(a) + w(b)$. The edge $\{a, b\}$ is removed, while the edges $\{a, h\}$ and $\{b, h\}$ that are incident to both a and b are merged to form a new edge $\{a+b, h\}$ in G_1 with edge weight $w(\{a+b, h\}) = w(\{a, h\})+w(\{b, h\}) = 2$. The same operations are performed to merge vertices c and d, e and f. After merging all the vertices involved in the edges of M, the remaining vertices h and g which are not incident to any edge of M are simply copied to G_1, completing the new coarsened graph G_1.

2.3 Uncoarsening procedure

In principle, the uncoarsening phase performs the opposite operations of the coarsening phase and successively recovers the intermediate graphs G_i $(i = m, m-1, \ldots, 0)$ in the reverse order of their creations. To recover G_i from G_{i+1}, each merged vertex of G_{i+1} is unfolded to obtain the original vertices and the associated edges of G_i. For an illustrative example, the same process applies as presented in Fig. 2 but with the reversed direction of each transition arrow. In practice, each corresponding intermediate graph recorded during coarsening is just restored.

For each recovered graph G_i, the partition of G_{i+1} is projected back to G_i and is further improved by the local refinement procedure. The uncoarsening process continues until the initial graph G_0 is recovered. At this point, IMSA terminates the current V-cycle and is ready to start the next V-cycle. Generally, the partition quality is progressively improved throughout the uncoarsening process. Precisely, the partition quality of G_i is usually better than that of G_{i+1} since there are more degrees of freedom for the local refinement procedure of an uncoarsened graph.

2.4 Local refinement with simulated annealing

For local refinement, IMSA mainly uses a dedicated simulated annealing procedure, which is complemented by an existing tabu search procedure.

2.4.1 Simulated annealing refinement procedure

Multilevel algorithms typically use pure descent algorithms for solution refinement at each level. This ensures fast convergence towards a local optimum that may be of mediocre quality compared to a global optimum. To reinforce the search capacity of the IMSA algorithm, the powerful simulated annealing method [23] is used, which has shown its effectiveness on the popular 2-way graph partitioning problem [20]. Moreover, a solution sampling strategy specially designed for MC-GPP (Section 2.4.2) is adopted to further ensure an effective examination of candidate solutions. To avoid search stagnation in a local optimum, the SA based refinement is complemented with a tabu search procedure (see Section 2.5).

The main scheme of the SA procedure is presented in Algorithm 2. Specifically, SA performs a number of search rounds (lines 3-21) with different temperature values to improve the current solution s (i.e., a cut $(S, \overline{S})$). At the start of each search round, the move counter mv is initialized to 0 (line 4). The procedure enters the 'for' loop to carry out $saIter$ iterations ($saIter$ is a parameter) with the current temperature T (initially set to T_0). At each iteration, SA randomly samples a neighbor solution s' of s from a set of constrained candidates (lines 5-19). Given a set of critical vertices $CV(s)$ (see Section 2.4.2), this is achieved by displacing a random vertex $v \in CV(s)$ from its current subset to the opposite subset (lines 6-7, where $s' \leftarrow s \oplus \mathit{Relocate}(v)$ denotes this move operation). The new solution s' then replaces the current solution s according to the following probability (lines 8-15),

Algorithm 2 Simulated annealing based local refinement.

Require: Graph $G = (V, E)$, input solution s, initial temperature T_0, move counter mv, maximum number of iterations per temperatures $saIter$, cooling ratio θ, frozen state parameter ar.
Ensure: Best partition s^{best} found during the search.

```
1: $T \leftarrow T_0$
2: $s^{best} \leftarrow s$
3: repeat
4:    $mv \leftarrow 0$
5:    for $iter = 1$ to $saIter$ do
6:       $v \leftarrow$ a random vertex from $CV(s)$
7:       $s' \leftarrow s \oplus Relocate(v)$                    /* Section 2.4.2 */
8:       if $\delta(v) < 0$ then
9:          $s \leftarrow s'$
10:         $mv \leftarrow mv + 1$
11:      else
12:         With probability defined in Equation (3)         /* Section 2.4.1 */
13:         $s \leftarrow s'$
14:         $mv \leftarrow mv + 1$
15:      end if
16:      if $\Phi(s) < \Phi(s^{best})$ then
17:         $s^{best} \leftarrow s$
18:      end if
19:   end for
20:   $T \leftarrow T * \theta$                          /* Cooling down the temperature */
21: until Acceptance rate ($mv/saIter$) is below $ar$ for 5 consecutive rounds
22: return $s^{best}$
```

$$Pr\{s \leftarrow s'\} = \begin{cases} 1 & , \text{ if } \delta < 0 \\ e^{-\delta/T} & , \text{ if } \delta \geq 0 \end{cases} \tag{3}$$

where $\delta = \Phi(s') - \Phi(s)$ is the conductance variation (also called the move gain) of transitioning from s to s'.

A better neighbor solution s' in terms of the conductance ($\delta < 0$) is always accepted as the new current solution s. Otherwise ($\delta \geq 0$), the transition from s to s' takes place with probability $e^{-\delta/T}$. After each solution transition, the move counter mv is incremented. The best solution s^{best} is updated each time a better solution is found (lines 16-18). Once the number of sampled solutions reaches $saIter$, the temperature T is cooled down by a constant factor $\theta \in [0, 1]$ (θ is a parameter, line 20), and SA proceeds to the next search round with this lowered new temperature. The termination criterion (the frozen state) of SA is met when the acceptance rate becomes smaller than a threshold ar (ar is a parameter) for 5 consecutive search rounds, where the acceptance rate is defined as $mv/saIter$ (line 21). Upon the termination of the SA procedure, the procedure returns the best recorded solution s^{best} (line 22).

One critical issue for the SA procedure concerns the initial temperature T_0. While a high T_0 leads to acceptance of many deteriorating uphill moves, a low initial temperature will have the same impact as a pure descent procedure resulting in the search to easily become trapped in a local optimum. To identify a suitable initial temperature, a simple binary search is used to provide a tradeoff between these two extremes. Starting from an initial temperature range $T \in [1.0e\text{-}20, 1.0]$, T_0 is initialized with the median value from this range, i.e., $T_0 = (1.0 + 1.0e\text{-}20)/2$. If the last round of the search using the current value of T_0 resulted in an acceptance rate of 50%, the value of T_0 is left unchanged. Otherwise, the value for T_0 used in next round of the search is set to be the median value of the lower or the upper temperature range depending on whether the acceptance rate is higher or lower than 50%. This process continues until a suitable initial temperature T_0 is found for a given problem instance. The other parameters ($saIter$, θ, and ar) of SA are investigated in Section 4.4.

2.4.2 Solution sampling with critical vertices

To generate a new candidate solution s' from the current solution s, the popular *Relocate* operator is applied that displaces a vertex from its current set to the opposite set. To avoid the generation of non-promising candidate solutions, the set of critical vertices are identified with respect to the current solution s. Specifically, let $s = (S, \overline{S})$ be the current solution. Vertex $v \in V$ is called a critical vertex if v is the endpoint of an edge in the cut-set $cut(s)$ [29]. Let $CV(s)$ denote the set of all the critical vertices in the cut-set $cut(s)$. Then, *Relocate* only operates on the vertices of $CV(s)$. By constraining the *Relocate* operation to the critical vertices, the algorithm avoids sampling many non-promising candidate solutions, as demonstrated in [30].

To ensure a fast computation of the conductance variation of a neighbor solution generated by *Relocate*, a streamlined incremental evaluation technique [30] is adopted. Let $s = \{S, \overline{S})$ be the current solution, $s' = \{S', \overline{S'}\}$ be the neighbor solution after relocating a vertex v of s from S to $\overline{S}$. The conductance of s' can be evaluated in $\mathcal{O}(1)$ time by simply updating $|cut(s')|$, $vol(S')$, $vol(\overline{S'})$ by $|cut(s')| = |cut(s)| + deg_S(v) - deg_{\overline{S}}(v)$, $vol(S') = vol(S) - deg(v)$, and $vol(\overline{S'}) = vol(\overline{S}) + deg(v)$, where $deg_S(v)$ (resp. $deg_{\overline{S}}(v)$) is the number of vertices adjacent to v in S (resp. in $\overline{S}$). Moreover, for each vertex w adjacent to v, $deg_{S'}(w)$ and $deg_{\overline{S'}}(w)$ can be updated in $\mathcal{O}(1)$ time by $deg_{S'}(w) = deg_S(w) - 1$ and $deg_{\overline{S'}}(w) = deg_{\overline{S}}(w) + 1$.

2.5 Additional solution refinement with tabu search

To further reinforce the solution refinement and go beyond the local optimum attained by the SA procedure, the constrained neighborhood tabu search (CNTS) procedure of [30] is additionally applied. CNTS relies on the same constrained neighborhood defined by the critical vertices and employs a dynamic tabu list to explore additional local optima. CNTS requires two parameters: the maximum number of consecutive non-improving iterations of tabu search D, and the tabu tenure management factor α. Section 4.4 explains the procedure used to tune these parameters and analyzes their sensitivity. More details about CNTS can be found in [30]. The use of CNTS is based on the preliminary observation that it can provide performance gains especially for the class of relatively small-sized instances with vertices less than one million (such as 'ok2010', 'va2010', 'nc2010', etc. See Appendix A, Tables A.2–A.4). Generally, the CNTS component can be safely disabled for large graphs without impacting the performance of the IMSA algorithm. In terms of solution refinement, CNTS is more focused on search intensification, as opposed to the more stochastic and diversified SA procedure. In practice, CNTS plays a complementary and secondary role by performing a much shorter search (determined by the parameter D) than SA during a round of IMSA.

2.6 Complexity of the proposed algorithm

The time complexity of the local refinement procedure is first considered. The SA procedure (Section 2.4) performs $saIter$ iterations. At each iteration, SA first identifies the set of critical vertices $CV(s)$, which can be achieved in $\mathcal{O}(|V| \times deg_{max})$ time where deg_{max} is the maximum degree of the graph. A candidate solution is then sampled in $\mathcal{O}(1)$ time. When a neighboring solution $s' = \{S', \overline{S'}\}$ is obtained, the conductance variation is calculated in $\mathcal{O}(1)$ time. Moreover, $deg_{S'}(w)$ and $deg_{\overline{S'}}(w)$ are updated in $\mathcal{O}(deg(v))$ time. Thus, SA requires $\mathcal{O}(saIter \times |V| \times deg_{max})$ time. For the CNTS procedure (Section 2.5), it can be achieved in $\mathcal{O}(D \times |V| \times deg_{max)}$ time [30]. Since the dominating part of the local refinement is the SA procedure, the time complexity of local refinement is $\mathcal{O}(saIter \times |V| \times deg_{max})$.

The whole IMSA algorithm (Algorithm 1) is composed of a series of V-cycles. For each V-cycle, the solution-guided coarsening procedure with the HEM heuristic requires $\mathcal{O}(m \times |E|)$ time, where m is the number of levels of each V-cycle. At each level, the partition of the intermediate graph is improved by the local refinement procedure with a time complexity of $\mathcal{O}(saIter \times |V| \times deg_{max})$. The uncoarsening procedure requires $\mathcal{O}(1)$ time, because it simply restores each corresponding intermediate graph recorded during the coarsening process, followed by the local refinement procedure. Consequently, the total time complexity of one V-cycle of the IMSA algorithm is bounded by $\mathcal{O}(m \times saIter \times |V| \times deg_{max})$.

The space complexity of IMSA can be evaluated as follows. First, IMSA represents a graph with an adjacency list whose space complexity is $\mathcal{O}(|V|+|E|)$. For each level, IMSA keeps an auxiliary graph, requiring $\mathcal{O}(m \times |E|)$ space for a V-cycle. Moreover, IMSA uses a table to maintain the sum of degrees of each vertex of G with $\mathcal{O}(m \times |V|)$ space. Therefore, the total space complexity of the IMSA algorithm for each V-cycle is given by $\mathcal{O}(m \times (|V| + |E|))$.

In practice, as long as the graph has a low or very low density, the space requirement is approximately linear in $|V|$. On the contrary, for dense and very dense (near-complete) graphs, the space requirement becomes quadratic in $|V|$.

3 Computational studies

This section is dedicated to a computational assessment of the proposed IMSA algorithm based on various benchmark instances.

3.1 Benchmark instances

The assessment was based on 66 very large benchmark instances with 54,870 to 23,947,347 vertices. The first 56 instances are very large graphs from *The 10th DIMACS Implementation Challenge Benchmark* [1], which were introduced for graph partitioning and graph clustering [4]. The remaining 10 instances are large real-world network graphs [32] from *The Network Data Repository online* [2]. These 66 instances are divided into three sets: Small set (24 instances, $|V| < 500,000$), Medium set (25 instances, $|V| \in [500,000,\ 5,000,000]$, and Large set (17 instances, $|V| > 5,000,000$). Among these 66 instances, 29 DIMACS graphs and 10 massive network graphs have previously been used for experimental evaluations in [29]. Additional 27 large-scale DIMACS benchmark graphs (with 114,599 to 21,198,119 vertices) were considered to better assess the scalability of the multilevel algorithm.

[1] `https://www.cc.gatech.edu/dimacs10/downloads.shtml`
[2] `http://networkrepository.com/index.php`

3.2 Experimental setting and reference algorithms

The proposed IMSA algorithm was implemented in C++ [3] and compiled using the g++ compiler with the "-O3" option. All the experiments were conducted on a computer with an AMD Opteron 4184 processor (2.8GHz) and 32GB RAM under Linux operating system. This computer was previously used to perform experimental evaluations reported in [29]. IMSA being a sequential algorithm, it was run on a single core of the AMD Opteron processor, as in [29].

Table 1 shows the setting of the IMSA parameters, which can be considered as the default setting of the algorithm. The procedure to tune these parameters is explained in Section 4.4. For meaningful assessments, this default setting was consistently used throughout all the experiments presented in this work. As shown in Section 3.3, IMSA achieves highly competitive results with this unique setting. Generally, the parameters can be fined-tuned to obtain improved results. Such a practice is useful when one seeks the best possible solution for a given graph.

To evaluate the performance of IMSA, comparisons are performed against three best performing MC-GPP algorithms from the literature: the two most recent metaheuristic algorithms (i.e., the hybrid evolutionary algorithm MAMC [29] and the breakout local search algorithm SaBTS [30]), as well as the popular and powerful max-flow algorithm MQI [24]. To ensure a fair comparison, all the compared algorithms were run on the same computing platform mentioned above with their default parameters setting. Each algorithm was independently executed 20 times per instance with a cutoff time of 60 minutes per run. Exactly like SaBTS and MAMC, each run of IMSA was initialized with a seeding solution provided by the MQI algorithm.

Out of the 66 benchmark instances tested in this work, the results for 39 instances (29 DIMACS graphs and the 10 massive network graphs) for the three reference algorithms are available in [29]. Since these results were obtained following the same experimental protocol as in the current work, they are directly used for the comparative studies. The compared algorithms were only ran on the 27 remaining instances.

[3] The code of the IMSA algorithm will be made publicly available at: `http://www.info.univ-angers.fr/pub/hao/IMSA.html`

Table 1
The parameters setting of the IMSA algorithm.

Parameter	Section	Description	Value
ct	§2.2	Coarsening threshold	60,000
saIter	§2.4	Maximum number of iterations per temperature of SA search	200,000
θ	§2.4	Cooling ratio of SA search	0.98
ar	§2.4	Frozen state parameter of SA search	5%
D	§2.5	Maximum number of consecutive non-improving iterations of tabu search	10,000
α	§2.5	Tabu tenure management factor of tabu search	80

Table 2
Summary results reported by the IMSA algorithm and the three reference algorithms (MAMC [29], SaBTS [30], and MQI [24]) on the three sets of 66 benchmark instances.

Benchmark set (size)	Algorithm pair	Indicator	#Wins	#Ties	#Losses	*p*-value
Small set (24)	**IMSA** *vs.* MAMC [29]	Φ_{best}	9	11	4	6.35e-01
		Φ_{avg}	12	8	4	1.63e-01
	IMSA *vs.* SaBTS [30]	Φ_{best}	13	9	2	5.54e-02
		Φ_{avg}	23	0	1	2.55e-04
	IMSA *vs.* MQI [24]	Φ_{best}	14	10	0	1.22e-04
		Φ_{avg}	24	0	0	1.82e-05
Medium set (25)	**IMSA** *vs.* MAMC [29]	Φ_{best}	17	7	1	1.18e-03
		Φ_{avg}	21	3	1	1.55e-04
	IMSA *vs.* SaBTS [30]	Φ_{best}	17	8	0	2.93e-04
		Φ_{avg}	25	0	0	1.23e-05
	IMSA *vs.* MQI [24]	Φ_{best}	18	7	0	1.96e-04
		Φ_{avg}	25	0	0	1.23e-05
Large set (17)	**IMSA** *vs.* MAMC [29]	Φ_{best}	16	1	0	4.37e-04
		Φ_{avg}	17	0	0	2.93e-04
	IMSA *vs.* SaBTS [30]	Φ_{best}	16	1	0	4.38e-04
		Φ_{avg}	16	0	1	3.84e-04
	IMSA *vs.* MQI [24]	Φ_{best}	16	1	0	4.38e-04
		Φ_{avg}	17	0	0	2.93e-04
∗Benchmark total (66)	**IMSA** *vs.* MAMC [29]	Φ_{best}	42	19	5	3.85e-05
		Φ_{avg}	50	11	5	3.07e-07
	IMSA *vs.* SaBTS [30]	Φ_{best}	46	18	2	3.26e-07
		Φ_{avg}	64	0	2	2.81e-11
	IMSA *vs.* MQI [24]	Φ_{best}	48	18	0	1.63e-09
		Φ_{avg}	66	0	0	1.64e-12

Table 3
The geometric means of the best and the average conductance values (G_{best} and G_{avg}) reported by the IMSA algorithm, and the three reference algorithms (MAMC [29], SaBTS [30], and MQI [24]) on the three sets of 66 instances.

Algorithm	Indicator	Benchmark set			∗Benchmark total
		Small set	Medium set	Large set	
IMSA	G_{best}	0.00102027	**0.00034119**	**0.00002953**	**0.00027054**
	G_{avg}	**0.00102543**	**0.00037545**	**0.00003197**	**0.00028685**
MAMC [29]	G_{best}	**0.00101896**	0.00034970	0.00003359	0.00028216
	G_{avg}	0.00103098	0.00045095	0.00010023	0.00041351
SaBTS [30]	G_{best}	0.00104022	0.00035199	0.00003360	0.00028502
	G_{avg}	0.00124996	0.00044828	0.00003888	0.00034674
MQI [24]	G_{best}	0.00103574	0.00035305	0.00003368	0.00028507
	G_{avg}	0.00125648	0.00044982	0.00003906	0.00034826

3.3 Computational results and comparisons with state-of-the-art algorithms

This section presents the computational results of the proposed IMSA algorithm, together with the results of the three reference algorithms (MAMC, SaBTS, and MQI) on the three sets of 66 benchmark instances.

Table 2 summarizes the overall comparison while the detailed results are listed in Appendix A (Tables A.2–A.4). Additionally, a global comparison is provided using the geometric mean metric [16] in Table 3 (the smaller the better). Section 4.1 shows a time-to-target analysis to investigate the computational efficiency of the compared algorithms.

In Table 2, column 1 shows the benchmark sets with the number of instances in parenthesis (Benchmark set (size)). Column 2 indicates the pair of the compared algorithms (Algorithm pair). Column 3 provides the quality indicators (Indicator) in terms of the best and the average conductance values (Φ_{best} and Φ_{avg}). Columns 4-6 show the number of instances on which IMSA reached a better (#Wins), worse (#Losses), or equal results (#Ties) in terms of each quality indicator. Furthermore, the last column provides the p-value of the Wilcoxon signed-rank test with a confidence level of 99% to assess whether there exists a statistically significant difference between IMSA and each reference algorithm in terms of the best and the average performances.

Table 3 shows the geometric means of each algorithm using the best and the average conductance values for the three sets of 66 instances (G_{best} and G_{avg}). The first two columns show the algorithms and the quality indicators (G_{best} and G_{avg}). Columns 3-5 present the G_{best} (or G_{avg}) results for each benchmark set, while the last column reports the overall G_{best} (or G_{avg}) results for all the

66 instances. The best values of G_{best} (or G_{avg}) across all the algorithms (the smaller the better) are highlighted in boldface.

From Tables 2, 3 and Tables A.2–A.4, the following observations can be made.

1) IMSA competes very favorably with the best existing MC-GPP algorithms in terms of solution quality, by reaching better results for more than 63% cases compared to all the reference algorithms considered jointly. Remarkably, even IMSA's average results are much better than the best results of the reference algorithms in most cases. The dominance of IMSA over the reference algorithms is better shown on medium and large instances, by reporting the best results for all but one or two instances in terms of Φ_{best} and Φ_{avg}. The small geometric means of IMSA further confirm its superiority over the reference algorithms.
2) IMSA is computational effective compared to the reference algorithms. Thanks to its multilevel strategy, IMSA requires similar or shorter time to find equal or better solutions for a number of medium and large graphs. The time-to-target analysis of Section 4.1 provides more evidences.
3) Relating the results of Tables A.2–A.4 and the main features of the tested graphs shown in Table A.1, it can be concluded that IMSA is particularly suitable for massive and sparse graphs, while its performance decreases on dense graphs. This behavior is consistent with the general multilevel optimization methods for graph partitioning.

This comparative study shows that the IMSA algorithm is a valuable tool for partitioning large and sparse graphs in complement with existing methods. To understand its behavior and functioning. Section 4 shows experiments to shed light on the contributions of the key algorithmic components.

4 Analysis

This section presents experiments to get insights into the influences of the components of the IMSA algorithm: iterated multilevel framework, simulated annealing local refinement, and parameters.

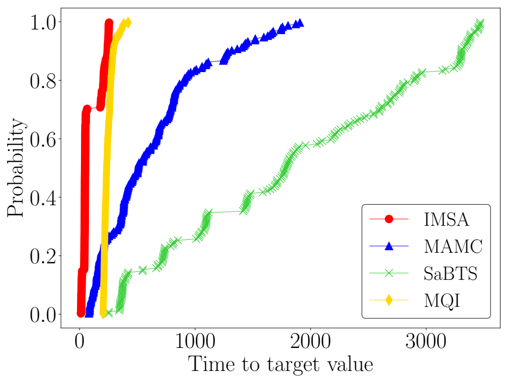
4.1 A time-to-target analysis of the compared algorithms

To investigate the computational efficiency of the compared algorithms: IMSA, MAMC [29], SaBTS [30], and MQI [24], a time-to-target (TTT) analysis [1] is performed. This study uses visual TTT plots to illustrate the running time distributions of the compared algorithms. Specifically, the Y-axis of the TTT plots displays the probability that an algorithm will find a solution that is at least as good as a given target value within a given run time, shown on the X-axis. The TTT plots are produced as follows. Each compared algorithm is independently run E_x times on each instance. For each of the E_x runs, the run time to reach a given target objective value is recorded. For each instance, the run times are then sorted in an ascending order. The i-th sorted run time t_i is associated with a probability $p_i = i/E_x$, and plotted as point (t_i, p_i), for $i = 1, \ldots, E_x$. This TTT experiment was based on 4 representative instances: *sc-pkustk13* ($|V| = 94,893$), *ga2010* ($|V| = 291,086$), *NLR* ($|V| = 4,163,763$), *delaunay_n24* ($|V| = 16,777,216$) with 200 independent runs per algorithm and per instance (i.e., $E_x = 200$). To make sure that the compared algorithms are able to reach the target objective value in each run, the target value was set to be 1% larger than the best objective value found by MQI. The results of this experiment are shown in Fig. 3.

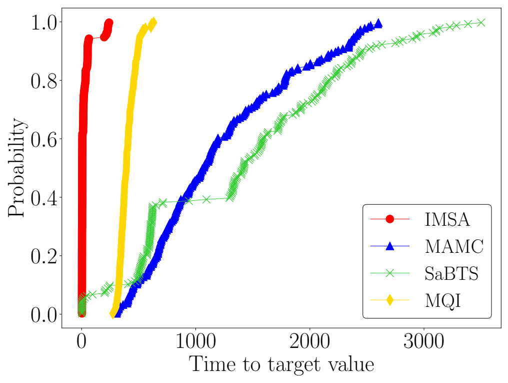
Fig. 3 clearly indicates that the proposed IMSA algorithm is always faster in attaining the given target value than the reference algorithms. The probability for IMSA to reach the target objective value within the first 100 to 200 seconds is around 90%, while MAMC, SaBTS, and MQI require much more times to attain the same result (at least 2500 seconds for the two large *NLR* and *delaunay_n24* graphs). One notices from Tables A.2–A.4 that MAMC and SaBTS show a shorter computation time for a number of instances, but they report typically worse results (especially on the medium and the large instances) than the IMSA algorithm. This indicates that IMSA can take full advantage of the allotted time budget to find solutions of better quality and avoid premature convergence. This is particularly true for large graphs. This experiment demonstrates that the IMSA algorithm is much more time efficient than the reference algorithms.

4.2 Usefulness of the iterated multilevel framework

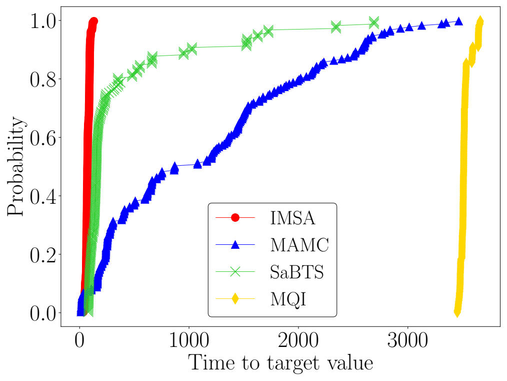
This section assesses the usefulness of the iterated multilevel framework. For this purpose, an IMSA variant called $(\text{SA+TS})^{restart}$ was created where the multilevel component was removed while keeping only the refinement procedure. To ensure a fair comparison, $(\text{SA+TS})^{restart}$ was performed in a multi-start way, until the cutoff time t_{max} (60 minutes) was reached. This experiment was conducted on 20 representative instances of a reasonable size and difficulty: *preferentialAttachment, smallworld, delaunay_n16, delaunay_n17, delaunay_n18, delaunay_n19, ga2010, oh2010, tx2010, wing, 144, ecology2, ecology1, thermal2, kkt_power, NACA0015, M6, AS365, luxembourg, belgium.* Each algorithm was independently run 20 times per instance with a cutoff time of 60 minutes per run. As a supplement, the p-values were also computed from the Wilcoxon signed-rank test in terms of the best and the average results.



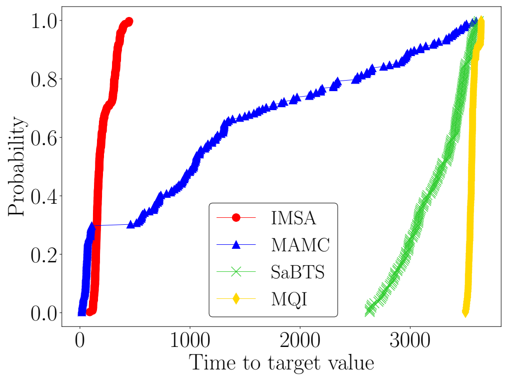
(a) *sc-pkustk13* ($|V| = 94,893$)



(b) *ga2010* ($|V| = 291,086$)



(c) *NLR* ($|V| = 4,163,763$)



(d) *delaunay_n24* ($|V| = 16,777,216$)

Fig. 3. Probability distributions of the run time (in seconds) needed to attain a given target objective value by each algorithm on the 4 representative instances.

Fig. 4 shows the best/average conductance gap between the two algorithms on these instances. The X-axis indicates the instance label (numbered from 1 to 20), while the Y-axis shows the best/average conductance gap in percentage, calculated as $(\Phi_{\mathcal{A}} - \Phi_{\text{IMSA}})/\Phi_{\text{IMSA}} \times 100\%$, where $\Phi_{\mathcal{A}}$ and Φ_{IMSA} are the best/average conductance values of (SA+TS)^{restart} and IMSA, respectively.

As observed in Fig. 4, IMSA clearly dominates (SA+TS)^{restart} in terms of the best (p-value = 1.32e-04) and the average (p-value = 1.03e-04) conductance values for the 20 instances. This experiment confirms the usefulness of the iterated multilevel framework, which positively contributes to the high performance of the algorithm.

4.3 Benefit of the SA-based local refinement

To evaluate the benefit of the SA local refinement procedure to the performance of the IMSA algorithm, an IMSA variant (denoted by IMSA^{descent}) was created where the SA procedure was replaced by a pure descent procedure

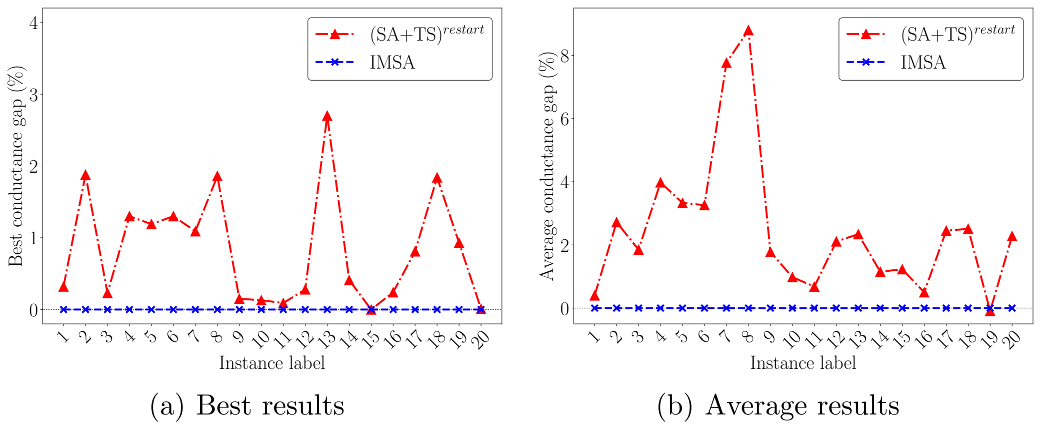


Fig. 4. Comparisons of the IMSA algorithm with a multi-start simulated annealing plus tabu search (denoted by (SA+TS)^{restart}) on the 20 representative instances.

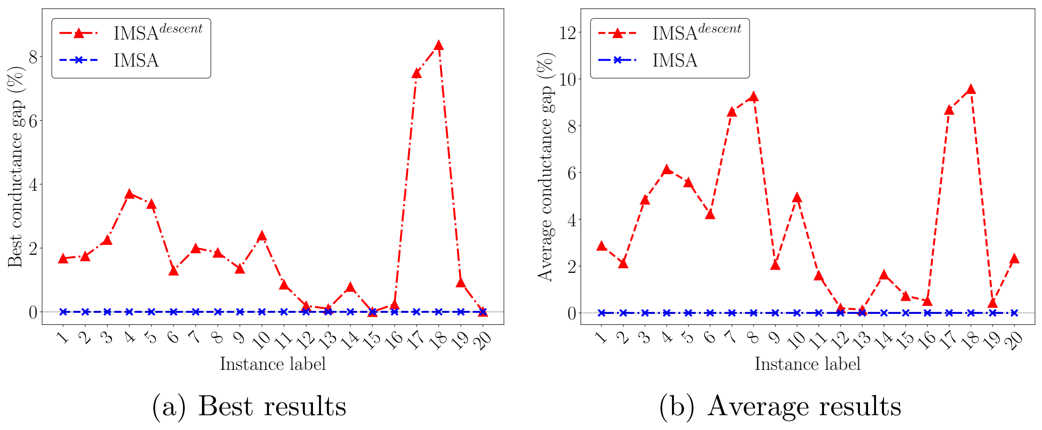
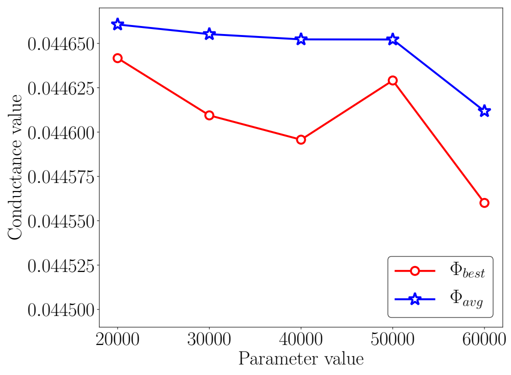


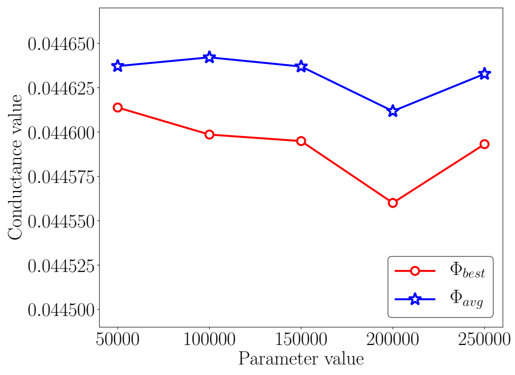
Fig. 5. Comparisons of the IMSA algorithm with an IMSA variant where the SA-based local refinement is replaced by a descent search (denoted by IMSA^{descent}) on the 20 representative instances.

using the best-improvement strategy. This experiment relies on the same 20 instances used in Section 4.2 and reports the same information. Fig. 5 plots the best/average conductance gap between the two algorithms on these instances. The X-axis indicates the instance label, while the Y-axis shows the best/average conductance gap in percentage.

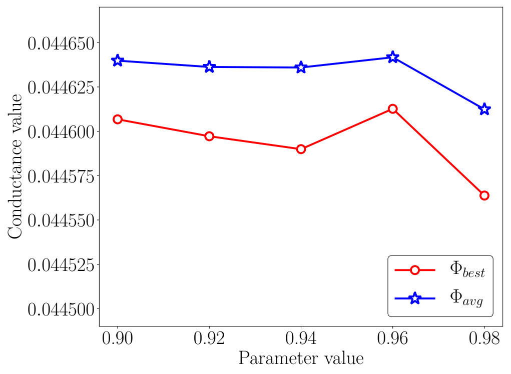
Fig. 5 shows that IMSA^{descent} reports significantly worse results in terms of both the best (p-value = 1.32e-04) and the average (p-value = 8.86e-05) conductance values for all the instances. This indicates that the SA procedure is the key element that ensures the high performance of IMSA and disabling it greatly deteriorates the performance.



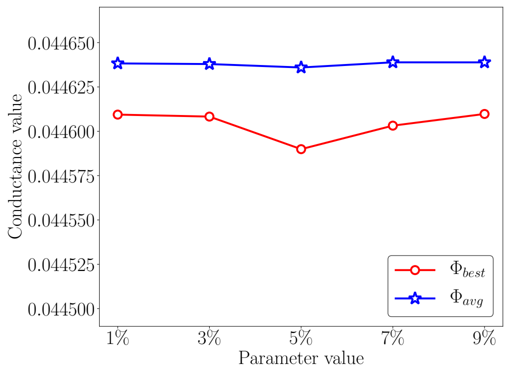
(a) Parameter *ct*



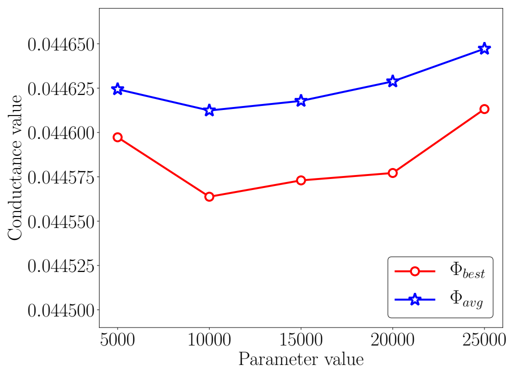
(b) Parameter $saIter$



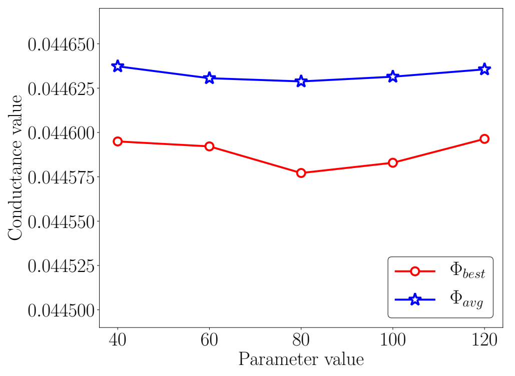
(c) Parameter θ



(d) Parameter *ar*



(e) Parameter D



(f) Parameter α

Fig. 6. Analysis of the parameters (ct, $saIter$, θ, ar, D, α) on its performance of the proposed IMSA algorithm.

4.4 Analysis of the parameters

The proposed IMSA algorithm requires six parameters: ct, $saIter$, θ, ar, D, and α. ct denotes the coarsening threshold in the solution-guided coarsening phase. $saIter$, θ, and ar are the three parameters related to the SA local refinement, where $saIter$ is the maximum number of iterations per temperature,

θ is the cooling ratio, and ar is the frozen state parameter. D and α are the maximum number of consecutive non-improving iterations of tabu search and the tabu tenure management factor, respectively.

To study the effect of these parameters on the performance of IMSA and to determine the most suitable setting for these parameters, a one-at-a-time sensitivity analysis [15] was performed as follows. For each parameter, a range of possible values were tested, while fixing the other parameters to their default values from Table 1. Specifically, the following values were used: $ct \in$ [20000, 30000, 40000, 50000, 60000], $saIter \in$ [50000, 100000, 150000, 200000, 250000], $\theta \in$ [0.90, 0.92, 0.94, 0.96, 0.98], $ar \in$ [1%, 3%, 5%, 7%, 9%], $D \in$ [5000, 10000, 15000, 20000, 25000], and $\alpha \in$ [40, 60, 80, 100, 120]. This experiment was conducted on 9 representative instances from the set of instances used in Sections 4.2 and 4.3, and based on 20 independent runs per parameter value with a cutoff time of 60 minutes per run. Fig. 6 shows the average values of Φ_{best} and Φ_{avg} obtained for the 9 instances, where the X-axis indicates the values of each parameter and the Y-axis shows the best/average conductance values over the 9 representative instances: *preferentialAttachment, smallworld, delaunay_n18, delaunay_n19, ga2010, oh2010, wing, 144, thermal2.*

Fig. 6 shows the impact of each parameter on the performance of IMSA. Specifically, for the parameter *ct*, $ct = 60000$ yields the best results for both Φ_{best} and Φ_{avg}. The default value of *ct* was set to 60000 in this study. For $saIter$, the value of 200000 is the best choice while a larger or a smaller value weakens the performance of IMSA. For the parameter θ, IMSA obtains the best performance with the value of 0.98 while smaller values decrease its performance. Furthermore, $ar = 5\%$ appears to be the best choice for IMSA. For the parameters D and α, the value 10000 and 80 were adopted respectively as their default values according to Fig. 6. The default values of all the parameters are summarized in Table 1.

5 Conclusion and future work

The iterated multilevel simulated annealing algorithm presented in this work is the first multilevel algorithm for the challenging NP-hard minimum conductance graph partitioning problem. Based on the general (iterated) multilevel framework, IMSA integrates a novel solution-guided coarsening method to construct a hierarchy of reduced graphs and a powerful simulated annealing local refinement procedure that makes full use of a constrained neighborhood to rapidly and effectively improve the quality of sampled solutions.

The performance of IMSA was assessed on three sets of 66 benchmark instances from the literature, including 56 graphs from the 10th DIMACS Implementation Challenge and 10 graphs from the Network Data Repository online, with up to 23 million vertices. Computational results demonstrated the competitiveness of the algorithm in finding high-quality partitions for large-scale sparse graphs. This work proves for the first time the value of the general multilevel approach for conductance graph partitioning.

From the application perspective, MC-GPP is a general and powerful graph model able to formulate a variety of real problems related to community detection, clustering, bioinformatics, computer vision, and large graph size estimation. Consequently, researchers and practitioners working on these real world problems can benefit from the proposed approach to find improved solutions. The availability of the source code of the algorithm will further facilitate such applications.

For future research, several directions could be followed. First, parallel computations are known to be quite useful for large graph partitioning (e.g., parallel graph partitioning for complex networks [31], distributed evolutionary partitioning [41] and distributed local search for partitioning large social networks [41]). Parallel algorithms were also integrated in popular partition packages such as ParMETIS (`http://glaros.dtc.umn.edu/gkhome/views/metis`) and KaHIP (`https://kahip.github.io/`). It would be highly relevant to create parallel versions of the IMSA algorithm to further increase its power. Second, population-based evolutionary algorithms have been successfully employed for solution refinement under the multilevel framework [5]. It would be interesting to study this approach for conductance partition. Third, the current IMSA implementation is more suitable for partitioning large sparse graphs than for dense graphs. It would be useful to investigate additional strategies to be able to handle both types of graphs. In particular, other graph representations using compact structure [14] may be considered to reduce the space complexity of the algorithm. Fourth, in addition to the studied memetic and local search methods in the literature, it is worthy investigating other metaheuristic-based algorithms to better handle various types of graphs and further enrich the MC-GPP toolkit. Finally, given the scarcity of exact approaches to MC-GPP in the existing literature, there is clearly a lot room for research in this direction.

Acknowledgment

The authors 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) (No.: 201606070096) for the first author is also acknowledged.

References

[1] R. M. Aiex, M. G. Resende, C. C. Ribeiro, TTT plots: a perl program to create time-to-target plots, Optimization Letters 1 (4) (2007) 355–366.

[2] R. Andersen, K. J. Lang, An algorithm for improving graph partitions, in: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2008, pp. 651–660.

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

[4] D. A. Bader, H. Meyerhenke, P. Sanders, D. Wagner, Graph partitioning and graph clustering, vol. 588, American Mathematical Society Providence, RI, 2013.

[5] U. Benlic, J. K. Hao, A multilevel memetic approach for improving graph k-partitions, IEEE Transactions on Evolutionary Computation 15 (5) (2011) 624–642.

[6] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Survey 35 (3) (2003) 268–308.

[7] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information Sciences 237 (2013) 82–117.

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

[9] J. Cheeger, A lower bound for the smallest eigenvalue of the laplacian, in: Proceedings of the Princeton Conference in honor of Professor S. Bochner, 1969, pp. 195–199.

[10] D. Cheng, R. Kannan, S. Vempala, G. Wang, A divide-and-merge methodology for clustering, ACM Transactions on Database Systems 31 (4) (2006) 1499–1525.

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

[12] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Company, New York, USA, 1979.

[13] M. Gendreau, J. Potvin (eds.), Handbook of Metaheuristics, vol. 146 of International Series in Operations Research & Management Science, Springer, 2010.

[14] F. Glaria, C. Hernández, S. Ladra, G. Navarro, L. Salinas, Compact structure for sparse undirected graphs based on a clique graph partition, Information Sciences 544 (2021) 485–499.

[15] D. Hamby, A review of techniques for parameter sensitivity analysis of environmental models, Environmental Monitoring and Assessment 32 (2) (1994) 135–154.

[16] S. Hauck, G. Borriello, An evaluation of bipartitioning techniques, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 16 (8) (1997) 849–866.

[17] B. Hendrickson, R. W. Leland, A multi-level algorithm for partitioning graphs., in: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing, 1995, p. 28.

[18] D. S. Hochbaum, Polynomial time algorithms for ratio regions and a variant of normalized cut, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (5) (2009) 889–898.

[19] D. S. Hochbaum, 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) (2013) 184–198.

[20] D. S. Johnson, C. R. Aragon, L. A. McGeoch, C. Schevon, Optimization by simulated annealing: An experimental evaluation; part I, graph partitioning, Operations Research 37 (6) (1989) 865–892.

[21] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on Scientific Computing 20 (1) (1998) 359–392.

[22] G. Karypis, V. Kumar, Metis 5.1.0: Unstructured graphs partitioning and sparse matrix ordering system, in: Technical Report, Department of Computer Science, University of Minnesota, 1998.

[23] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.

[24] K. Lang, S. Rao, A flow-based method for improving the expansion or conductance of graph cuts, in: International Conference on Integer Programming and Combinatorial Optimization, Springer, 2004, pp. 325–337.

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

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

[27] Y. Lim, W. J. Lee, H. J. Choi, U. Kang, MTP: discovering high quality partitions in real world graphs, World Wide Web 20 (3) (2017) 491–514.

[28] J. Lu, H. Wang, Uniform random sampling not recommended for large graph size estimation, Information Sciences 421 (2017) 136–153.

[29] Z. Lu, J. K. Hao, Q. Wu, A hybrid evolutionary algorithm for finding low conductance of large graphs, Future Generation Computer Systems 106 (2020) 105–120.

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

[31] H. Meyerhenke, P. Sanders, C. Schulz, Parallel graph partitioning for complex networks, IEEE Transactions on Parallel and Distributed Systems 28 (9) (2017) 2625–2638.

[32] R. A. Rossi, N. K. Ahmed, The network data repository with interactive graph analytics and visualization, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI Press, 2015, pp. 4292–4293.

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

[34] J. Šíma, S. E. Schaeffer, On the NP-completeness of some graph cluster measures, in: International Conference on Current Trends in Theory and Practice of Computer Science, Springer, 2006, pp. 530–537.

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

[36] U. Trottenberg, C. W. Oosterlee, A. Schüller, Multigrid, Academic Press, London, 2001.

[37] A. Valejo, V. Ferreira, R. Fabbri, M. C. F. d. Oliveira, A. d. A. Lopes, A critical survey of the multilevel method in complex networks, ACM Computing Surveys 53 (2) (2020) 1–35.

[38] T. Van Laarhoven, E. Marchiori, Local network community detection with continuous optimization of conductance and weighted kernel k-means, Journal of Machine Learning Research 17 (1) (2016) 5148–5175.

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

[40] C. Walshaw, Multilevel refinement for combinatorial optimisation problems, Annals of Operations Research 131 (1-4) (2004) 325–372.

[41] B. Zheng, L. Ouyang, J. Li, Y. Lin, C. Chang, B. Li, T. Chen, H. Peng, Towards a distributed local-search approach for partitioning large-scale social networks, Information Sciences 508 (2020) 200–213.

[42] Z. A. Zhu, S. Lattanzi, V. Mirrokni, A local algorithm for finding well-connected clusters, in: International Conference on Machine Learning, 2013, pp. 396–404.

A Appendix

This appendix reports the detailed results of the proposed IMSA algorithm and the three reference algorithms (MAMC [29], SaBTS [30], and MQI [24]) on the three sets of 66 benchmark instances whose main features are shown in Table A.1.

According to the experimental protocol of Section 3.2, each algorithm was run with its default parameters setting and executed 20 times per instance with a cutoff time of 60 minutes per run. The results of 39 small and medium instances (29 DIMACS graphs and the 10 massive network graphs) for the three reference algorithms were extracted from [29]. The results of the compared algorithms on the remaining instances were obtained with the above protocol. Tables A.2–A.4 show the computational results of the compared algorithms.

In Tables A.2–A.4, columns 1 and 2 indicate the name and the number of vertices $|V|$ for each instance. The remaining columns show the results reached by IMSA and the reference algorithms (MAMC, SaBTS, and MQI): the best conductance (Φ_{best}), the average conductance (Φ_{avg}), the number of times Φ_{best} was reached across 20 independent runs (*hit*), and the average computation time per run in seconds to reach the final solution ($t(s)$). The best of the Φ_{best} (or Φ_{avg}) values among all the compared algorithms for each instance is highlighted in boldface. An asterisk ($*$) indicates a strictly best solution among all the results. Bold and asterisked values thus correspond to the best upper bounds for the associated graphs.

Note that it is meaningful to compare the computation times of two algorithms on a graph only if they report the same objective value. Given that IMSA reached better results on many instances, the timing information is provided for indicative purposes. For a meaningful comparison of computational efficiency of the algorithms, the reader is referred to the time-to-target analysis shown in Section 4.1.

Table A.1
Main features of the 66 benchmark instances from “The 10th DIMACS Implementation Challenge Benchmark” and “The Network Data Repository online”.

Instance	$\|V\|$	$\|E\|$	Density
Small set (24)			
sc-nasasrb	54,870	1,311,227	8.71e-04
wing	62,032	121,544	6.32e-05
delaunay_n16	65,536	196,575	9.15e-05
sc-pkustk13	94,893	3,260,967	7.24e-04
preferentialAttachment	100,000	499,985	1.00e-04
smallworld	100,000	499,998	1.00e-04
luxembourg	114,599	119,666	1.82e-05
delaunay_n17	131,072	393,176	4.58e-05
144	144,649	1,074,393	1.03e-04
web-arabic-2005	163,598	1,747,269	1.31e-04
soc-gowalla	196,591	950,327	4.92e-05
delaunay_n18	262,144	786,396	2.29e-05
ok2010	269,118	637,074	1.76e-05
va2010	285,762	701,064	1.72e-05
nc2010	288,987	708,310	1.70e-05
ga2010	291,086	709,028	1.67e-05
cnr-2000	325,557	2,738,969	5.17e-05
mi2010	329,885	789,045	1.45e-05
mo2010	343,565	828,284	1.40e-05
oh2010	365,344	884,120	1.32e-05
soc-twitter-follows	404,719	713,319	8.71e-06
pa2010	421,545	1,029,231	1.16e-05
il2010	451,554	1,082,232	1.06e-05
soc-youtube	495,957	1,936,748	1.57e-05
Medium set (25)			
soc-flickr	513,969	3,190,452	2.42e-05
delaunay_n19	524,288	1,572,823	1.14e-05
ca-coauthors-dblp	540,486	15,245,729	1.04e-04
soc-FourSquare	639,014	3,214,986	1.57e-05
eu-2005	862,664	16,138,468	4.34e-05
tx2010	914,231	2,228,136	5.33e-06
ecology2	999,999	1,997,996	4.00e-06
ecology1	1,000,000	1,998,000	4.00e-06
NACA0015	1,039,183	3,114,818	5.77e-06
delaunay_n20	1,048,576	3,145,686	5.72e-06
inf-roadNet-PA	1,087,562	1,541,514	2.61e-06
thermal2	1,227,087	3,676,134	4.88e-06
belgium	1,441,295	1,549,970	1.49e-06
G3_circuit	1,585,478	3,037,674	2.42e-06
kkt_power	2,063,494	6,482,320	3.04e-06
delaunay_n21	2,097,152	6,291,408	2.86e-06
netherlands	2,216,688	2,441,238	9.94e-07
M6	3,501,776	10,501,936	1.71e-06
333SP	3,712,815	11,108,633	1.61e-06
AS365	3,799,275	11,368,076	1.58e-06
venturiLevel3	4,026,819	8,054,237	9.93e-07
NLR	4,163,763	12,487,976	1.44e-06
delaunay_n22	4,194,304	12,582,869	1.43e-06
hugetrace00	4,588,484	6,879,133	6.53e-07
channel	4,802,000	42,681,372	3.70e-06
Large set (17)			
hugetric00	5,824,554	8,733,523	5.15e-07
hugetric10	6,592,765	9,885,854	4.55e-07
italy	6,686,493	7,013,978	3.14e-07
adaptive	6,815,744	13,624,320	5.87e-07
hugetric20	7,122,792	10,680,777	4.21e-07
great-britain	7,733,822	8,156,517	2.73e-07
delaunay_n23	8,388,608	25,165,784	7.15e-07
germany	11,548,845	12,369,181	1.85e-07
asia	11,950,757	12,711,603	1.78e-07
hugetrace10	12,057,441	18,082,179	2.49e-07
road_central	14,081,816	16,933,413	1.71e-07
hugetrace20	16,002,413	23,998,813	1.87e-07
delaunay_n24	16,777,216	50,331,601	3.58e-07
hugebubbles00	18,318,143	27,470,081	1.64e-07
hugebubbles10	19,458,087	29,179,764	1.54e-07
hugebubbles20	21,198,119	31,790,179	1.41e-07
road_usa	23,947,347	28,854,312	1.01e-07

Table A.2: Computational results on the Small set of 24 benchmark instances obtained by the IMSA algorithm and the three reference algorithms (MAMC [29], SaBTS [30], and MQI [24]).

Instance	$\|V\|$	**IMSA**				MAMC [29]				SaBTS [30]				MQI [24]			
		Φ_{best}	Φ_{avg}	*hit*	$t(s)$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$
sc-nasasrb	54870	**0.00325489**	**0.00325489**	20/20	12	**0.00325489**	**0.00325489**	20/20	15	**0.00325489**	0.00334089	2/20	2728	**0.00325489**	0.00335273	2/20	89
wing	62032	***0.00651615**	**0.00652917**	3/20	1564	0.00654906	0.00662078	1/20	1953	0.00668219	0.00689348	1/20	0	0.00668630	0.00690103	1/20	26
delaunay_n16	65536	***0.00224851**	**0.00225366**	1/20	1321	0.00226376	0.00229371	1/20	1292	0.00232641	0.00239002	1/20	0	0.00233083	0.00239444	1/20	37
sc-pkustk13	94893	0.00906291	0.00908081	1/20	1830	***0.00905805**	**0.00906634**	15/20	1730	0.00916933	0.00946360	1/20	1638	0.00921128	0.00952533	1/20	193
preferential Attachment	100000	0.29464260	0.29536141	1/20	2973	***0.28717318**	**0.28804940**	1/20	2886	0.29457883	0.29584725	1/20	2264	0.31132997	0.33039853	1/20	4
smallworld	100000	0.10377462	0.10464450	1/20	3251	***0.09948259**	**0.10321531**	1/20	2739	0.10105860	0.10387462	1/20	3181	0.11322141	0.11604861	1/20	9
luxembourg	114599	***0.00013710**	0.00014745	1/20	469	0.00013832	**0.00014707**	1/20	1299	0.00013837	0.00014830	1/20	0	0.00013846	0.00014875	1/20	379
delaunay_n17	131072	***0.00155911**	**0.00156350**	1/20	1740	0.00156857	0.00161819	1/20	2000	0.00161683	0.00166666	1/20	0	0.00161703	0.00166800	1/20	109
144	144649	***0.00604341**	**0.00605368**	1/20	1412	0.00605305	0.00607374	1/20	2690	0.00610660	0.00620893	1/20	206	0.00611994	0.00622039	1/20	99
web-arabic-2005	163598	**0.00000306**	**0.00000306**	20/20	9	**0.00000306**	**0.00000306**	20/20	1	0.00000406	0.00000408	16/20	3189	**0.00000306**	0.00000323	10/20	28
soc-gowalla	196591	**0.01234567**	**0.01234567**	20/20	9	**0.01234567**	**0.01234567**	20/20	0	**0.01234567**	0.01247394	16/20	3286	**0.01234567**	0.01247394	16/20	15
delaunay_n18	262144	***0.00109995**	**0.00110789**	1/20	2079	0.00113332	0.00114381	1/20	762	0.00113724	0.00117144	1/20	1	0.00113731	0.00117238	1/20	251
ok2010	269118	**0.00046030**	**0.00046030**	20/20	49	**0.00046030**	0.00046677	2/20	501	0.00046031	0.00050597	1/20	192	0.00046031	0.00050643	1/20	236
va2010	285762	**0.00050629**	**0.00050629**	20/20	3	**0.00050629**	0.00050810	17/20	3	**0.00050629**	0.00054526	6/20	129	**0.00050629**	0.00054958	4/20	287
nc2010	288987	***0.00029077**	**0.00029125**	1/20	50	0.00029079	0.00029215	1/20	1218	0.00029129	0.00030381	1/20	79	0.00029130	0.00030411	1/20	267
ga2010	291086	***0.00035772**	**0.00036104**	1/20	1731	0.00035935	0.00036353	1/20	1083	0.00036488	0.00039204	1/20	445	0.00036488	0.00039249	1/20	224
cnr-2000	325557	**0.00000156**	**0.00000156**	20/20	9	**0.00000156**	**0.00000156**	20/20	1	**0.00000156**	0.00000649	12/20	0	**0.00000156**	0.00000649	12/20	24
mi2010	329885	**0.00009247**	**0.00009247**	20/20	4	**0.00009247**	**0.00009247**	20/20	1	**0.00009247**	0.00020506	13/20	72	**0.00009247**	0.00022270	12/20	217
mo2010	343565	**0.00044010**	**0.00044010**	20/20	5	**0.00044010**	0.00044334	13/20	481	**0.00044010**	0.00050980	1/20	82	**0.00044010**	0.00051012	1/20	310
oh2010	365344	0.00046194	**0.00046598**	1/20	152	***0.00045030**	0.00047809	1/20	312	0.00047052	0.00050873	1/20	116	0.00047052	0.00050917	1/20	358
soc-twitter-follows	404719	**0.00540540**	**0.00540540**	20/20	4	**0.00540540**	**0.00540540**	20/20	38	**0.00540540**	0.01515060	1/20	1260	**0.00540540**	0.01522024	1/20	19
pa2010	421545	**0.00026616**	**0.00026616**	20/20	5	**0.00026616**	**0.00026616**	20/20	2	**0.00026616**	0.00027731	11/20	0	**0.00026616**	0.00028228	10/20	398
il2010	451554	***0.00024986**	**0.00024986**	20/20	57	0.00024997	0.00025260	2/20	515	0.00025204	0.00026736	1/20	5	0.00025204	0.00026746	1/20	436
soc-youtube	495957	**0.00892857**	**0.00892857**	20/20	12	**0.00892857**	**0.00892857**	20/20	2	**0.00892857**	0.01134011	12/20	550	**0.00892857**	0.01134011	12/20	58

Table A.3: Computational results on the Medium set of 25 benchmark instances obtained by the IMSA algorithm and the three reference algorithms (MAMC [29], SaBTS [30], and MQI [24]).

Instance	$\lvert V\rvert$	**IMSA**				MAMC [29]				SaBTS [30]				MQI [24]			
		Φ_{best}	Φ_{avg}	*hit*	$t(s)$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$
soc-flickr	513969	**0.00444444**	**0.00444444**	20/20	14	**0.00444444**	**0.00444444**	20/20	1	**0.00444444**	0.00565198	13/20	658	**0.00444444**	0.00565198	13/20	43
delaunay_n19	524288	***0.00077949**	**0.00078464**	1/20	1710	0.00078693	0.00080017	1/20	729	0.00078757	0.00081717	1/20	47	0.00078966	0.00081794	1/20	1807
ca-coauthors-dblp	540486	**0.00226757**	**0.00226757**	20/20	17	**0.00226757**	**0.00226757**	20/20	5	**0.00226757**	0.00244841	15/20	108	**0.00226757**	0.00244841	15/20	221
soc-FourSquare	639014	***0.24627988**	**0.25827142**	1/20	1878	0.26530612	0.26530612	20/20	4	0.26530612	0.26564625	15/20	743	0.26530612	0.26564625	15/20	40
eu-2005	862664	**0.00004930**	**0.00004930**	20/20	21	**0.00004930**	**0.00004930**	20/20	11	**0.00004930**	0.00064963	5/20	1219	**0.00004930**	0.00064963	5/20	224
tx2010	914231	***0.00023852**	0.00024769	1/20	519	0.00024221	**0.00024274**	2/20	176	0.00024258	0.00025289	1/20	2692	0.00024258	0.00025289	1/20	2386
ecology2	999999	***0.00050250**	**0.00051256**	1/20	2040	0.00050254	0.00051514	1/20	2661	0.00051407	0.00054807	1/20	186	0.00051588	0.00054822	1/20	3601
ecology1	1000000	***0.00050202**	**0.00051180**	1/20	1742	0.00050250	0.00051874	2/20	2634	0.00052769	0.00055124	1/20	548	0.00052775	0.00055158	1/20	3600
NACA0015	1039183	***0.00025909**	**0.00025990**	1/20	1591	0.00025970	0.00026124	2/20	0	0.00025970	0.00026124	2/20	153	0.00025970	0.00026124	2/20	3601
delaunay_n20	1048576	***0.00054264**	**0.00054980**	1/20	1925	0.00055891	0.00056128	9/20	27	0.00055890	0.00057389	1/20	997	0.00055891	0.00057390	1/20	3601
inf-roadNet-PA	1087562	**0.00008221**	**0.00008221**	20/20	57	**0.00008221**	0.00008230	15/20	4	**0.00008221**	0.00008789	2/20	418	**0.00008221**	0.00008789	2/20	3593
thermal2	1227087	***0.00025097**	**0.00025213**	1/20	1851	0.00025296	0.00025339	10/20	223	0.00025296	0.00025687	1/20	864	0.00025296	0.00025687	1/20	3601
belgium	1441295	**0.00004606**	**0.00004910**	1/20	541	**0.00004606**	0.00005024	1/20	1975	**0.00004606**	0.00005031	1/20	1	**0.00004606**	0.00005042	1/20	3601
G3_circuit	1585478	***0.00035679**	**0.00036099**	1/20	232	0.00036328	0.00036393	8/20	335	0.00036328	0.00038075	1/20	741	0.00036328	0.00038076	1/20	3595
kkt_power	2063494	**0.00161233**	**0.00232759**	3/20	599	0.00161236	0.00303129	1/20	1038	**0.00161233**	0.00235664	3/20	6	0.00161236	0.00235796	1/20	667
delaunay_n21	2097152	***0.00039276**	**0.00039276**	20/20	63	0.00039293	0.00039397	8/20	45	0.00039293	0.00041812	1/20	1854	0.00039293	0.00041813	1/20	3603
netherlands	2216688	**0.00001098**	**0.00002176**	2/20	1510	**0.00001098**	0.00002581	1/20	970	**0.00001098**	0.00002583	2/20	2	**0.00001098**	0.00002590	1/20	3602
M6	3501776	***0.00024210**	**0.00024312**	1/20	2547	0.00025614	0.00026321	1/20	1296	0.00026023	0.00026687	1/20	14	0.00026485	0.00027112	1/20	3605
333SP	3712815	**0.00003860**	**0.00010253**	2/20	1665	**0.00003860**	0.00012072	2/20	731	**0.00003860**	0.00012183	2/20	11	**0.00003860**	0.00012378	2/20	3604
AS365	3799275	***0.00019836**	**0.00019953**	1/20	1805	0.00021155	0.00021682	1/20	1203	0.00021490	0.00022072	1/20	11	0.00021721	0.00022384	1/20	3605
venturiLevel3	4026819	***0.00006604**	**0.00006936**	1/20	1670	0.00007586	0.00007914	1/20	1386	0.00007623	0.00008008	1/20	10	0.00007649	0.00008057	1/20	3603
NLR	4163763	***0.00029019**	**0.00029361**	1/20	2025	0.00030997	0.00032223	1/20	1089	0.00031270	0.00032705	1/20	18	0.00031758	0.00033050	1/20	3605
delaunay_n22	4194304	***0.00027769**	**0.00028008**	1/20	2082	0.00031352	0.00031504	13/20	286	0.00031352	0.00032530	1/20	2272	0.00031352	0.00032532	1/20	3605
hugetrace00	4588484	***0.00009883**	**0.00010148**	1/20	1525	0.00010064	0.00010504	1/20	1869	0.00009973	0.00010383	1/20	9	0.00010064	0.00010504	1/20	3604
channel	4802000	0.00113916	**0.00117489**	1/20	2118	***0.00113065**	0.03289337	1/20	2947	0.00119290	0.00124485	1/20	24	0.00120730	0.00125040	1/20	3610

Table A.4: Computational results on the Large set of 17 benchmark instances obtained by the IMSA algorithm and the three reference algorithms (MAMC [29], SaBTS [30], and MQI [24]).

Instance	$\|V\|$	**IMSA**				MAMC [29]				SaBTS [30]				MQI [24]			
		Φ_{best}	Φ_{avg}	*hit*	$t(s)$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$	Φ_{best}	Φ_{avg}	*hit*	$t(s)$
hugetric00	5824554	***0.00013290**	**0.00014285**	1/20	2067	0.00014325	0.00015247	1/20	0	0.00014278	0.00015124	1/20	9	0.00014325	0.00015247	1/20	3606
hugetric10	6592765	***0.00010251**	**0.00010503**	1/20	2167	0.00010885	0.00011206	1/20	0	0.00010783	0.00011095	1/20	8	0.00010885	0.00011206	1/20	3607
italy	6686493	***0.00000529**	**0.00000630**	1/20	1863	0.00000641	0.00000878	1/20	41	0.00000641	0.00000878	1/20	0	0.00000641	0.00000878	1/20	3605
adaptive	6815744	***0.00011538**	**0.00011735**	1/20	2010	0.00012195	0.00013036	1/20	1499	0.00013152	0.00013919	1/20	7	0.00013183	0.00013953	1/20	3607
hugetric20	7122792	***0.00009821**	**0.00010044**	1/20	2166	0.00010252	0.00010689	1/20	0	0.00010195	0.00010586	1/20	8	0.00010252	0.00010689	1/20	3607
great-britain	7733822	***0.00001090**	**0.00001245**	1/20	1689	0.00001302	0.00001524	1/20	689	0.00001302	0.00001526	1/20	0	0.00001302	0.00001526	1/20	3607
delaunay_n23	8388608	***0.00019856**	**0.00020019**	1/20	2403	0.00022443	0.00022552	8/20	526	0.00022443	0.00023219	1/20	2906	0.00022443	0.00023220	1/20	3611
germany	11548845	***0.00000937**	**0.00001000**	1/20	1940	0.00001032	0.00001218	1/20	603	0.00001032	0.00001219	1/20	0	0.00001032	0.00001219	1/20	3609
asia	11950757	**0.00000055**	**0.00000067**	7/20	820	**0.00000055**	0.00000100	7/20	39	**0.00000055**	0.00000122	7/20	0	**0.00000055**	0.00000122	2/20	3608
hugetrace10	12057441	***0.00006080**	**0.00007145**	1/20	2408	0.00007548	0.00082926	1/20	720	0.00007515	0.00008142	1/20	11	0.00007548	0.00008207	1/20	3609
road_central	14081816	***0.00000720**	**0.00000823**	1/20	2424	0.00001000	0.00001128	9/20	272	0.00001000	0.00001399	1/20	3240	0.00001000	0.00001399	1/20	3620
hugetrace20	16002413	***0.00004908**	**0.00005035**	1/20	1686	0.00005071	0.00151422	1/20	1260	0.00004996	0.00005140	1/20	7	0.00005017	0.00005174	1/20	3613
delaunay_n24	16777216	***0.00014110**	**0.00014330**	1/20	2975	0.00016180	0.00016208	11/20	1256	0.00016180	0.00016509	1/20	3420	0.00016180	0.00016509	1/20	3621
hugebubbles00	18318143	***0.00005012**	0.00005232	2/20	1303	0.00005027	0.00192212	1/20	1800	0.00005027	**0.00005177**	1/20	9	0.00005027	0.00005272	1/20	3616
hugebubbles10	19458087	***0.00005081**	**0.00005259**	1/20	2354	0.00005350	0.00206087	1/20	1800	0.00005315	0.00005587	1/20	17	0.00005356	0.00005640	1/20	3621
hugebubbles20	21198119	***0.00004257**	**0.00004398**	1/20	1641	0.00004369	0.00215594	1/20	1800	0.00004259	0.00004469	1/20	11	0.00004278	0.00004490	1/20	3620
road_usa	23947347	***0.00000329**	**0.00000398**	1/20	3034	0.00000584	0.00000606	11/20	493	0.00000584	0.00000743	1/20	3420	0.00000584	0.00000743	1/20	3625