Knowledge-Based Systems 151 (2018) 45–61

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Multilevel approach for combinatorial optimization in bipartite network

Alan Valejo*, Maria Cristina Ferreira de Oliveira, Geraldo P.R. Filho, Alneu de Andrade Lopes

Institute of Mathematical and Computer Sciences (ICMC), University of São Paulo (USP), P.O. Box 668, São Carlos, SP P.C. 14560-970, Brazil

A R T I C L E   I N F O

Article history:
Received 12 July 2017
Revised 12 March 2018
Accepted 13 March 2018
Available online 20 March 2018

Keywords:
Complex networks
Bipartite networks
Combinatorial optimization
Meta-heuristic
Multilevel optimization
Large-scale networks

A B S T R A C T

Multilevel approaches aim at reducing the cost of a target algorithm over a given network by applying it to a coarsened (or reduced) version of the original network. They have been successfully employed in a variety of problems, most notably community detection. However, current solutions are not directly applicable to bipartite networks and the literature lacks studies that illustrate their application for solving multilevel optimization problems in such networks. This article addresses this gap and introduces a multilevel optimization approach for bipartite networks and the implementation of a general multilevel framework including novel algorithms for coarsening and uncorsening, applicable to a variety of problems. We analyze how the proposed multilevel strategy affects the topological features of bipartite networks and show that a controlled coarsening strategy can preserve properties such as degree and clustering coefficient centralities. The applicability of the general framework is illustrated in two optimization problems, one for solving the Barber's modularity for community detection and the second for dimensionality reduction in text classification. We show that the solutions thus obtained are statistically equivalent, regarding accuracy, to those of conventional approaches, whilst requiring considerably lower execution times.

1. Introduction

Bipartite networks comprise a particular class of network models in which the set of vertices is split into two disjoint subsets, with edges connecting vertices placed in different sets. Also known as two-layer networks, they provide a powerful representation of relationships in many real-world systems, including document-word [1], protein-ligand [2], actor-movie [3], georeferenced user-location [4] and paper co-authorship or citation networks [5]. Bipartite network models have been widely employed in hard combinatorial optimization problems that require finding a minimum (or maximum) cost, wherein the number of possible states is finite and usually exponential. Many such problems, e.g., biclique, matching, vertex cover, community structure, traveling salesman and network coloring [6] have proven to be $\mathcal{NP}$-complete or $\mathcal{NP}$-hard.

Multilevel techniques are being investigated as a global strategy to handle decision-making and optimization problems in a variety of application domains. We refer the reader interested in applications to management problems to recent literature on the topic [7–9]. In this paper we investigate the multilevel strategy to handle computationally expensive optimization problems in bipartite networks. In this context, the approach consists of iteratively coarsening an original network into a hierarchy of smaller sized approximations. A starting solution is obtained in the coarsest network and successively projected back and refined over the inverse sequence of coarsened networks, until the original one. Previous studies demonstrated the strategy enables running computationally expensive algorithms on large networks with no significant loss in solution quality [10–16]. Sciences [17] and Walshaw [18] argued over the relevance and feasibility of the multilevel strategy for solving combinatorial optimization problems. Empirical evidence has been shown by Walshaw [18] that the coarsening process filters the solution space by gradually removing irrelevant high-cost solutions and drastically reducing the search space, and hence, optimization convergence times.

Multilevel algorithms have been applied to many classic network problems, including network coloring [19], traveling salesman [20], network drawing [19], network partitioning [21] and computation of centrality measures [22]. However, current multilevel approaches are not directly applicable to bipartite networks and, to the best of our knowledge, the multilevel strategy has not been considered in this context.

* Corresponding author.
E-mail addresses: alan@icmc.usp.br (A. Valejo), cristina@icmc.usp.br (M.C. Ferreira de Oliveira), alneu@icmc.usp.br (A.d.A. Lopes).

https://doi.org/10.1016/j.knosys.2018.03.021
0950-7051/© 2018 Elsevier B.V. All rights reserved.

We address this gap and introduce a novel multilevel optimization approach applicable to bipartite networks. Furthermore, we describe an implementation of this approach as a general-purpose multilevel framework that incorporates two novel efficient matching algorithms, as well as novel contracting and uncoarsening algorithms.

In order to illustrate its potential, we employed the framework to handle two distinct problems defined in bipartite networks, namely community detection and dimensionality reduction. In the community detection problem, tests on a large set of synthetic networks demonstrated that, combined with a proper local search strategy, it yields good speedups and preserves solution quality. When employed to perform dimensionality reduction in text classification it yielded encouraging results in terms of both runtime and accuracy as compared with a standard dimensionality reduction technique.

The remainder of the paper is organized as follows: Section 2 reviews some basic concepts on networks and provides an overview of the standard multilevel approach. Section 3 discusses previous work and application of multilevel strategies on combinatorial optimization problems. Section 4 introduces a multilevel formulation for bipartite networks and its implementation. Section 5 reports our empirical results, which include (i) an analysis of how the multilevel representation impacts the topological features of a real bipartite network; (ii) an empirical study of instantiating the framework to solve the community detection problem on a large synthetic test suite; and (iii) an empirical study of its application to dimensionality reduction in text classification. Section 6 briefly discusses how the general framework can be tuned for application in other types of problems. Finally, Section 7 summarizes our findings and discusses future work.

2. Background

2.1. Basic concepts

Let $G = (V, E, \sigma, \omega)$ be an undirected weighted network, where $V = \{1, \dots, n\}$ denotes the set of vertices and $E \subseteq V x V$ denotes the set of edges, such that $(v, u) = \{(u, v) = (v, u) \mid u, v \in V\}$. Let $n = |V|$ be the total number of vertices and $m = |E|$ be the total number of edges, where operator "|.|" stands for the cardinality of a set. The weight of an edge (u, v) is represented by $\omega(u, v)$ with $\omega : VxV \to \mathbb{R}^*$ and the weight of a vertex v is represented by $\sigma(v)$ with $\sigma : V \to \mathbb{R}^*$.

A network $G = (V, E, \sigma, \omega)$ is bipartite (two-layer network) if V is partitioned into two sets V_1 and V_2, such that $V_1 \cap V_2 = \emptyset$ and $E \subseteq V_1 \, x \, V_2$. Hereafter, each vertex subset is called a layer. A bipartite network thus has two layers so that vertices in the same layer are not connected.

The degree of a vertex $v \in V$, denoted κ_v, is given by the total weight of its adjacent edges, i.e. $\kappa_v = \sum_{u \in V} w(v, u)$. The h-hop neighborhood of v, denoted $\Gamma_h(v)$, is formally defined as the vertices in set $\Gamma_h(v) = \{u \mid$ there is a path of length h between v and $u\}$. Thus, the 1-hop neighborhood of v, $\Gamma_1(v)$, is the set of vertices adjacent to v; the 2-hop neighborhood, $\Gamma_2(v)$, is the set of vertices 2-hops away from v, and so forth.

A similarity score $S(u, v)$ can be computed from a pair of vertices u and v. A fundamental structural similarity function between a pair of vertices is given by the number of common neighbors, defined as $S_{cn}(u, v) = |\Lambda(u, v)|, \ \Lambda(u, v) = \{\Gamma_1(u) \cap \Gamma_1(v)\}$. Alternatively, a weighted common neighbors similarity function can be defined by Eq. (1), where the term $log(1 + s(z))$ is used to prevent negative scores [23].

$$S_{wcn}(u, v) = \sum_{z \in \Lambda(u,v)} \frac{\omega(u, z) + \omega(v, z)}{log(1 + s(z))} \qquad s(u) = \sum_{z \in \Gamma_1(u)} \omega(u, z) \quad (1)$$

The local clustering coefficient of a vertex is given by the probability of its neighbors being connected [3,24]. This statistics is closely related to transitivity, which measures the relative frequency of triangles in the vertex neighborhood. The clustering coefficient is defined by Eq. (2):

$$cc(v) = \frac{2tr(v)}{|\Gamma_1(v)|(|\Gamma_1(v)| - 1)}, \quad (2)$$

where $tr(u)$ denotes the number of edges $(v, z) \in E$, such that v, $z \in \Gamma_1(u)$ and cc relies on the enumeration of the triangles in the network. However, as triangles do not occur in bipartite networks this definition is not valid. An equivalent metrics for bipartite graphs was introduced by Latapy et al. [25], defined in Eq. (3), which captures the overlap between vertex neighborhoods in the same layer.

$$cc_b(v, u) = \frac{|\Gamma_1(u) \cap \Gamma_1(v)|}{|\Gamma_1(u) \cup \Gamma_1(v)|} \qquad cc_b(v) = \frac{\sum_{v \in \Gamma_2(u)} cc_b(v, u)}{|\Gamma_2(u)|}, \quad (3)$$

2.2. General-purpose multilevel optimization

A multilevel optimization is formally defined as a meta-heuristics that combines different heuristics to guide, modify and possibly fix a solution obtained from a target algorithm (or operations of the subordinate heuristics, local search or global search) and refines this solution over multiple iterations. It operates in three phases, namely *coarsening, solution finding* and *uncoarsening*. In the coarsening phase, the network size is successively reduced to obtain coarser network representations; in the solution finding phase a starting solution is obtained applying the target algorithm in the coarsest representation; in the uncoarsening phase, the starting solution is successively projected back to the intermediate networks and refined, until obtaining the final solution.

Fig. 1 illustrates such a process, considering an initial network G_0 (in which the original problem instance is defined), where G_L denotes the coarsest network obtained after L coarsening steps (levels), S_L denotes the starting solution obtained in G_L, and S_0 denotes the final refined solution obtained in G_0.

2.2.1. Coarsening

The coarsening phase constructs a hierarchy of coarsened networks G_l from the initial network G_0, yielding intermediate network approximations on multiple levels-of-detail. The process requires two algorithms, namely *matching*, which defines which vertices will be merged, and *contracting*, which builds the reduced representation, given the matching. Let $G_l = (V_l, E_l, \sigma_l, \omega_l)$ be the network model coarsened at level l, with $|V_0| > |V_1| > \dots > |V_l|$.

Coarsening starts with the matching step. According to some given restriction, in general, pairs of vertices are selected for matching, producing a set of unordered pairs called *vertex matching, independent edge set* or simply *matching*. Formally, a matching M consists of a set of pairwise non-adjacent edges. Heavy-edge matching is a popular algorithm for this purpose, which attempts to find a matching of maximal weight [26].

At any level, the coarsening of a network must preserve its topological features, implying that vertex and edge weights of the reduced network must reflect the connectivity of its parent. This will be guaranteed by a proper choice of matching strategy, which is a key component of effective multilevel optimizations. A matching strategy inadequate to support the solution finding phase will impair the quality of the solution derived by a multilevel algorithm and its performance.

Once the matching is defined, a contracting algorithm constructs the coarsened network, by joining matched vertex pairs into a single *super-vertex* (sV). A child network G_{l+1} will inherit the non-joined vertices from its parent. In order for G_{i+1} to be a good proxy to its parent, given a super-vertex $sV = \{v, u\} \in V_{i+1}$ its

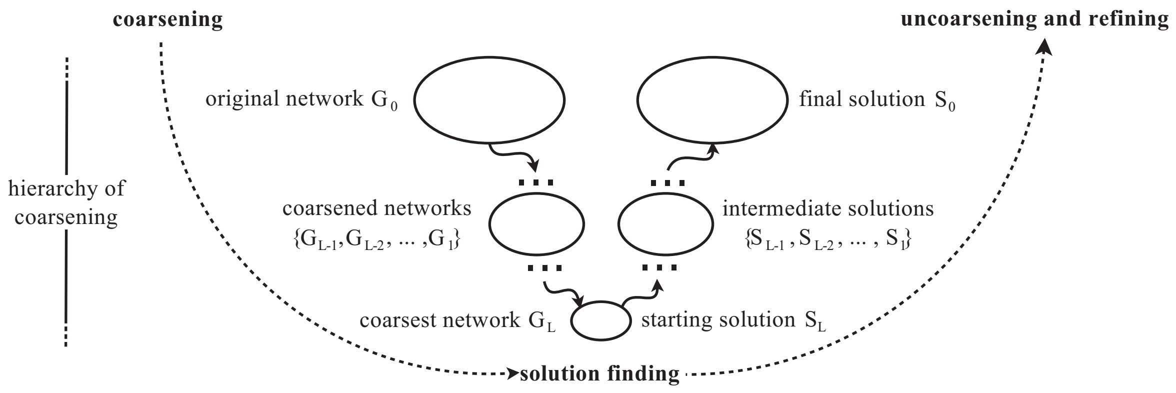


Fig. 1. Phases of a multilevel optimization process: coarsening, solution finding and uncoarsening.

weight $\sigma(sV)$ is computed as the sum of weights $\sigma(v)$ and $\sigma(u)$, $\{u, v\} \in V_i$. Furthermore, the edges incident to vertices $\{u, v\} \in V_i$ are joined to obtain the so-called super-edges incident to sV.

2.2.2. Solution finding

This phase employs the target algorithm to solve the problem on the smallest network G_l. Let S be the set of all possible solutions in the coarsest problem instance. Given an objective (cost) function $f : S \to \mathbb{R}$ (or $\mathbb{N}$) that assigns a cost to each solution in S, the aim is to find a state $s \in S$ with minimum (or maximum) cost. For instance, in the traveling salesman problem $f(s)$ expresses the length of tour s, whereas in a network community detection problem it denotes some measure of community quality. Since the coarsest network is possibly very small, it becomes feasible to employ computationally expensive target algorithms to find a starting solution [11].

2.2.3. Uncoarsening and refinement

The uncoarsening (also known as solution projection) phase successively transfers the solution available at a current level to the upper level in the hierarchy, i.e., the solution obtained in the coarsest network G_l is successively projected through intermediate networks $G_{l-1}, G_{l-2}, \ldots, G_1$ up to the original network G_0.

Solution S_l is constructed from S_{l+1} simply by assigning vertices $\{u, v\} \in V_l$ to the same set of their parent $sV \in V_{l+1}$. Although S_l is a local minima of f in G_l, this may not be the case of solution S_{l-1}, derived for the upper level G_{l-1}, with respect to G_i. Therefore, a refinement heuristics can be applied to avoid local minima and improve solution quality. Local operations can move the solution towards a lower cost neighboring solution in the search space; for instance, in a community detection problem vertices can be moved between adjacent communities to improve a target quality measure.

3. Related work

Early studies of multilevel optimization were mostly designed to speed up the recursive bisection problem, as its high computational cost prevents wider applicability [27]. One of the first theoretical analysis was presented by Karypis and Kumar [11], who demonstrated multilevel approaches can find high-quality communities in a variety of networks. Later, Karypis and Kumar [26] introduced the now widely adopted matching algorithms HEM (Heavy Edge Matching), LEM (Light Edge Matching) and MCH (Modified Edge Matching). Other studies relevant for the development and expansion of the multilevel approach were conducted by Walshaw and Cross [28], who presented a theoretical multilevel formulation of the Kernighan–Lin method for mesh partitioning, and by Korosec et al. [29], who introduced a new multilevel colony optimization applicable to several optimization problems. Sciences [17] and Walshaw [18] provided compelling evidence the multilevel framework is an extremely useful addition to combinatorial optimization toolkits, although it can not be considered a panacea.

Previous work on multilevel optimization in complex networks can be broadly organized into four categories. A first category encompasses studies that explore features of network domains; e.g., Abou–Rjeili and Karypis [30] studied scale-free networks (with power-law degree distribution), Oliveira and Seok [31] designed a multilevel spectral approach that explores features of biological networks (protein complexes) and Valejo et al. [14] considered properties of social networks, such as high transitivity and assortativity. A second group comprises contributions focused on scalability, e.g., investigating parallel and distributed paradigms that improve the performance of the coarsening and refinement phases [16,32–44]. A third category of studies is concerned with the optimization of a target objective function, e.g., several contributions focused on improvements in modularity [13,16,45–50]. Finally, there are significant contributions in applications, including (and not limited to) graph coloring [51], traveling salesman problem [20], graph drawing [19], biomedical feature selection [52], covering design [53], DNA sequencing [54], vehicle routing [55], semi-supervised learning [56], partitioning or community detection [21] and computation of centrality measures [22].

We are not aware of any previous effort concerned with the usage of multilevel strategies to solve optimization problems in bipartite networks, despite the relevance of this kind of network to real-world modeling problems. Current multilevel methods cannot be directly applied in bipartite models without adaptations, e.g., they do not consider vertex types, whereas the layers in a bipartite network usually represent distinct types of entities that must be handled independently. For the sake of illustration, suppose a text document collection modeled as a document-word bipartite network. For a start, matching vertices that represent words and vertices that represent documents (i.e., matching of vertices of different layers) would not be meaningful in most application scenarios. Moreover, as the number of words is typically much higher, coarsening the word layer may be sufficient to reduce the asymptotic convergence of a target algorithm.

4. A multilevel approach in bipartite networks

This section introduces a multilevel optimization framework designed to handle bipartite networks. Bearing in mind the previous discussion, we have defined two restrictions that establish the major distinction between current multilevel methods and the one proposed here:

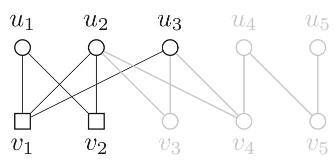


Fig. 2. Example of a bipartite network with $V_1 = \{u_1, u_2, u_3, u_4, u_5\}$ and $V_2 = \{v_1, v_2, v_3, v_4\}$.

1. Vertices are only allowed to match their set of two-hop neighbors.
2. The matching algorithm must operate on vertices of the same layer.

Such restrictions support cost-effective implementations of multilevel strategies in bipartite networks. From the definitions of bipartite networks and two-hop neighborhoods, the first restriction implies adjacent vertices are not matched. Furthermore, vertices can only match others in their two-hop neighborhood set, rather than any non-adjacent vertex. Considering, for instance, the network in Fig. 2, vertex u_1 can match vertices u_2 and u_3, which are non-adjacent and are in $\Gamma_2(u_1)$.

The first restriction alone does not enforce independent handling of layers, which is guaranteed by the second restriction, which states layers (either one of them or both) will be processed independently in the coarsening phase. Therefore, the coarsening will not match vertices of different kinds and distinct coarsening and refinement algorithms, or distinct parameterizations of the same algorithm may be adopted in processing each layer. Such a restriction also favors the adoption of distributed or parallel processing strategies.

Algorithm 1 summarizes the general ***m**ultilevel **o**ptimization framework for **b**ipartite networks* (*MOb*). Similarly to the standard multilevel approach, it comprises the phases of coarsening (lines 1–4), solution finding (line 5) and uncoarsening (lines 7–8). It takes as inputs the initial bipartite network $G = (V_1 \cup V_2, E, \sigma, \omega)$, and for each layer a maximum number of coarsening levels L and a layer reduction factor *rf*.

The coarsening is applied to each layer (line 1), level by level until the desired reduction factor is attained, by calls to a matching algorithm (line 3) and a contracting algorithm (line 4). The matching algorithm considers the above restrictions in selecting the vertex pairs to produce the list of independent edges (line 3). The reduction factor ($rf \in [0, 0.5]$) is multiplied by the number of vertices to determine the maximum matching number. If $rf = 0.5$ (the maximum reduction factor), each coarsening iteration will (potentially) reduce the number of vertices by a factor of two, yielding a logarithmic decrease in network size along the process. This is not guaranteed if the network is disconnected or highly sparse, since in this case there may be an insufficient number of edges. The call to the contracting algorithm in the next step (line 4) is responsible for creating the coarsened bipartite network, in which any matched vertex pairs have been merged into super-vertices.

The target algorithm is then executed in the coarsest network G_l to obtain a starting solution S_l (line 5). Finally, in the subsequent uncoarsening phase this solution is projected back, up to G_0, through the space of intermediate solutions $S_{l-1}, S_{l-2}, \ldots, S_1, S_0$ (lines 7–9), possibly refining the solutions at each level (line 8). Although we do not investigate refinement algorithms in this work, the rationale is to apply a local search strategy to improve the current solution, i.e., algorithms should explore small regions of the solution space, in order to reduce impact on performance and scalability. We do introduce novel algorithms for matching, contracting and uncoarsening.

4.1. Matching in bipartite networks

We propose a straightforward matching algorithm called ***r**andom **g**reedy **m**atching for **b**ipartite networks* (*RGMb*), described in Algorithm 2, where $S(u, v)$ denotes a similarity function.

A vertex u is picked randomly (Line 4) at each iteration, and as long as it has not yet been matched, one of its unmatched two-hop neighbors v so that $S(u, v)$ is maximal is chosen (Line 5). Vertices u and v are marked as matched and removed from the list of unmatched vertices (Line 7). The process iterates until no more vertices can be eliminated from the list. For unweighted edges, a

Algorithm 1. *MOb*: multilevel optimization framework for bipartite networks.

```
Input:
    bipartite network              : $G = (V, E, \sigma, \omega)$
    maximal number of levels       : array $L = \{L_i \mid L_i \in [0, n] \subset \mathbb{Z}\}$ with $1 \le |L| \le 2$
    reduction factor for each layer: array $rf = \{rf_i \mid rf_i \in (0, 0.5] \subset \mathbb{R}\}$ with $1 \le |rf| \le 2$
    layers to be coarsened         : array $layers = \{i \mid i \in \{1, 2\} \subset \mathbb{Z}\}$ with $1 \le |layers| \le 2$
Output:
    solution                       : S
1  for $i \in$ layers do
2      while $l \le L_i$ or layer i is not as small as desired do
3          $M \leftarrow$ matching($G_l$, $i$, $rf_i$);
4          $G_{l+1} \leftarrow$ contracting($G_l$, M);
5          increment $l$;
6  $S_l \leftarrow$ target_algorithm($G_l$);
7  while $l \ne 0$ do
8      $S_{l-1} \leftarrow$ uncoarsening($G_{l-1}$, $G_l$, $S_l$);
9      $S_{l-1} \leftarrow$ refining($G_{l-1}$, $S_{l-1}$);
10     decrement $l$;
Return: S
```

Algorithm 2. *RGMb*: random greedy matching for bipartite networks.

```
Input:
    bipartite network      : $G = (V, E, \sigma, \omega)$
    selected layer         : $l \in \{1, 2\}$
    reduction factor       : $rf \in (0, 0.5] \subset \mathbb{R}$
Output:
    matching               : array of tuples $M = \{(u, v) \mid u, v \in V\}$
1  matching $M \leftarrow \emptyset$;
2  merge count $mc \leftarrow rf * |V|$;
3  while $mc > 0$ do
4      randomly select a vertex $u \in V_l$;
5      select $v \in \Gamma_2(u)$ of maximal $S(u, v)$;
6      $M \leftarrow M \cup (u, v)$;
7      remove $u$ and $v$ from $V_l$;
8      decrement $mc$;
Return: M
```

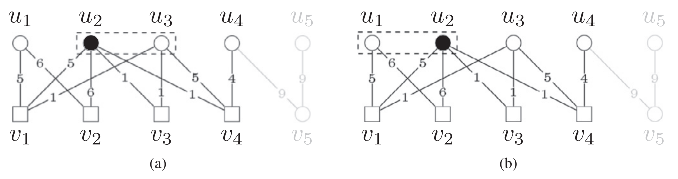


Fig. 3. Bipartite network, such that $V_1 = \{u_1, u_2, u_3, u_4, u_5\}$ and $V_2 = \{v_1, v_2, v_3, v_4\}$; (a) pairs $\{u_2, u_3\}$ included in the matching; (b) pairs $\{u_1, u_2\}$ included in the matching.

random neighbor is selected for matching, otherwise, the heaviest adjacent edge is selected.

One may consider different similarity functions, e.g., common neighbors similarity S_{cn} or weighted common neighbors similarity S_{wcn}. For illustration, consider a graph $G = (V_1 \cup V_2, E)$, shown in Fig. 3, with $V_1 = \{u_1, u_2, u_3, u_4, u_5\}$ and $V_2 = \{v_1, v_2, v_3, v_4, v_5\}$, $\Gamma_2(u_2) = \{u_1, u_3, u_4\}$ and u_1, u_2, u_3 and u_4 unmatched. Suppose vertex u_2 has been randomly chosen for matching. Adopting S_{cn} as the similarity function, then $S_{cn}(u_2, u_1) = 2$, $S_{cn}(u_2, u_3) = 3$ and $S_{cn}(u_2, u_4) = 1$, therefore, pair $\{u_2, u_3\}$ would be included in the matching, as illustrated in Fig. 3(a). Alternatively, for a choice of S_{wcn} as similarity function, $S_{wcn}(u_1, u_2) = 11$, $S_{wcn}(u_1, u_3) = 7$ and $S_{wcn}(u_2, u_4) = 2.5$ and pair $\{u_1, u_2\}$ would be included, as illustrated in Fig. 3(b).

Algorithm *RGMb* does not ensure an optimal choice over all possible matchings. An alternative strategy would choose from a list of vertex pairs sorted in decreasing order of similarity scores, which may be kept in an appropriate data structure, e.g., a heap or a priority queue. An alternative algorithm that implements this strategy is called ***g**reedy sorted **m**atching for **b**ipartite networks* (*GMb*) (Algorithm 3): it selects the best possible match for a vertex from its two-hop neighborhood (line 4).

GMb constructs a priority queue of potential matches ranked by similarity (Lines 3–5) and uses it to retrieve optimal matching choices (Line 8). In case of a tie, it makes a random choice, in order to favor further exploration of the solution space over multiple iterations. If a vertex is selected that has already been matched, it is skipped. *GMb*, albeit slower than its random search counterpart *RGMb*, is more robust and yields better performance (see Section 5).

For illustration, consider the graph $G = (V_1 \cup V_2, E)$ depicted in Fig. 4, with $V_1 = \{u_1, u_2, u_3, u_4, u_5\}$ and $V_2 = \{v_1, v_2, v_3, v_4, v_5\}$. Suppose the matching is being performed on layer V_1. Adopting the S_{cn} similarity, pairs $\{u_2, u_3\}$ and $\{u_4, u_5\}$ would be included in the matching, as illustrated in Fig. 4(a). Alternatively, for a choice of S_{wcn} similarity, pairs $\{u_1, u_2\}$ and $\{u_4, u_5\}$ would be included, as shown in Fig. 4(b).

Variations of the general matching algorithm can be obtained depending on the combined choices of matching strategy and similarity function $S(u, v)$. We report four variants, namely *RGMb*$_{cn}$ and *RGMb*$_{wcn}$ for a choice of random matching with, respectively, S_{cn} and S_{wcn} similarity; *GMb*$_{cn}$ and *GMb*$_{wcn}$ for the equivalent combinations with greedy matching.

4.2. Contracting in bipartite networks

Algorithm 4 (*Cb*, ***c**ontracting of a **b**ipartite network from a matching*) describes how to create a coarsened bipartite network from a matching M. It takes as input a bipartite network G_l and a corresponding matching M. First, vertex pairs $\{u, v\} \in M$ are mapped into a *successor vector* (lines 5–6) and joined into a single super-vertex $sV \in G_{l+1}$ (lines 4–9). The *successor vector* will be accessed in the uncoarsening phase to assign pair $\{u, v\} \in G_{l+1}$ to the same subset of its super-vertex sV. Any vertices not included in M are inherited by G_{l+1} (line 10). After the mapping, adjacent edges in E_l are joined and added to E_{l+1} (lines 11–18), i.e., each pair $(u, v) \in E_l$ holds its successors (w, z) and if edge (w, z) already exists in E_{l+1} its weight is increased by $\omega_l(u, v)$ (lines 15–16); otherwise, a new edge (w, z) with weight $\omega_l(u, v)$ is inserted into E_{l+1} (lines 17–18).

Fig. 5 illustrates the contracting process. Considering, for instance, the previous example, in which pairs $\{u_2, u_3\}$ and $\{u_4, u_5\}$ have been included in the matching, as illustrated in Fig. 5(a), the resulting coarsened network is depicted in Fig. 5(b).

The capability of handling layers with distinct parameter settings (rf and L) and coarsening algorithms is a compelling feature of the proposed framework. This is advantageous in many situations, since layers in many real network models are highly unbalanced in size, as in the already mentioned document-word networks. Coarsening just one of the layers is also a useful feature, for example, in dimensionality reduction problems, e.g., to reduce the space of words in document-word networks, as illustrated in Section 5.3.

Algorithm 3. *GMb*: greedy matching for bipartite networks.

```
Input:
    bipartite network      : $G = (V, E, \sigma, \omega)$
    selected layer         : $l \in \{1, 2\}$
    reduction factor       : $rf \in (0, 0.5] \subset \mathbb{R}$
Output:
    matching               : array of tuples $M = \{(u, v) \mid u, v \in V\}$

1  matching $M \leftarrow \emptyset$;
2  $Q \leftarrow$ max priority queue;
3  forall the $u \in V_l$ do
4      select $v \in \Gamma_2(u)$ of maximal $S(u, v)$;
5      insert $(u, v)$ into $Q$ with priority $S(u, v)$;
6  merge count $mc \leftarrow rf * |V|$;
7  while $mc > 0$ do
8      $(u, v) \leftarrow$ remove from $Q$;
9      if $u$ and $v$ unmatched then
10         $M \leftarrow M \cup (u, v)$;
11     decrement $mc$;
Return: $m$
```

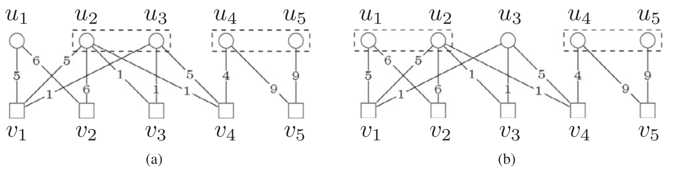


Fig. 4. Bipartite network, such that $V_1 = \{u_1, u_2, u_3, u_4, u_5\}$ and $V_2 = \{v_1, v_2, v_3, v_4\}$; (a) pairs $\{u_2, u_3\}$ and $\{u_4, u_5\}$ included in the matching; (b) pairs $\{u_1, u_2\}$ and $\{u_4, u_5\}$ included in the matching.

4.3. Uncoarsening algorithm

The solution finding phase executes the target algorithm to obtain a starting solution S_L. The representation of S_L depends on the problem being handled. In a community detection problem, it is described as a partitioning of the vertex set into non-empty partitions P_k with $\cup P_k = S_L$, $P_k \subseteq V_L$. Each iteration of the uncoarsening phase projects the current solution, obtained in network G_l, to network G_{l-1} as described in Algorithm 5 *Ub* (***u**ncoarsening of **b**ipartite networks*). The rationale is, for each vertex $u \in V_{l-1}$ in network G_{l-1} (line 2), to obtain its corresponding successor vertex $w \in V_l$ (line 3) and assign vertex u to its partition (lines 4–5). An implementation can keep the successor vector as a global variable, or it may be provided as a network attribute.

As discussed later (Section 6), the algorithm can be adapted with little effort to handle other types of problems and representations, e.g., in link prediction, S would be a set of predicted edges $E_P \notin E$. In some problems the solution finding and uncoarsening phases are omitted, e.g., in dimension reduction the coarsest bipartite network is itself the final solution (see Section 5.3).

4.4. Computational complexity

Consider the multilevel strategy applied to both layers of a network. Computing a matching requires, for each selected vertex u, to find its h-hop neighborhood $\Gamma_h(u)$, which takes $O(n\langle\kappa\rangle h)$ time, where $\langle\kappa\rangle$ is the average network degree and h specifies the number of distance hops. Most real networks are sparse, i.e., $\langle\kappa\rangle \ll n$ and $m \approx n$. This study only considers two-hop neighborhoods ($h = 2$), which leads to notable computational savings, since Γ_2 can be reached in nearly linear time. Appropriate data structures and parallelism can be employed to maximize the efficiency of the matching step [57–59]. The subsequent contracting step relies directly on the matching and can be very fast in sparse networks. A coarsened network can be mapped and built in time $O(|E|)$, since the process is iterated over all edges of its parent network.

If the coarsening parameter is set to its maximum value ($rf = 0.5$), at each iteration the current network size will be reduced at most by a factor of two relative to its parent. In this case, given $1 < L < n$, the coarsening steps would produce networks of sizes $(n, \frac{n}{2}, \ldots, \frac{n}{n})$, yielding a reduction factor $O(\log_2(n))$ over the L iterations. For instance, departing from a network with $n = 100$ vertices, $rf = 0.5$ and $L = 2$, potentially 50 vertex pairs will be merged in the first iteration (from G_0 to G_1) and 25 in the second (from G_1 to G_2), a reduction factor of 4. In most real-world problems, such a reduction factor would suffice to produce a manageable model to run the optimization algorithm. As the cost of coarsening decreases drastically at each level, the number of levels L may be neglected in the estimation of the computational cost.

The complexity of the solution finding phase is determined by the target algorithm. If T_{tg} is the cost of the target algorithm on a bipartite network with n vertices, and T_{ss} is the cost to obtain the starting solution, it is not too difficult to show that $T_{ss} = T_{tg}/(2^L)$. Insofar as the hierarchy of approximations produced by coarsening, $L = 1$ yields $(n, \frac{n}{2})$, $L = 2$ yields $(n, \frac{n}{2}, \frac{n}{4})$, $L = 3$ yields $(n, \frac{n}{2}, \frac{n}{4}, \frac{n}{8})$ and so on. Hence, the number of vertices in the coarsest network is approximately $\frac{n}{(2^L)}$ and therefore $T_{ss} = \frac{T_{tg}}{(2^L)}$. Parameter L can thus be chosen to control the cost deemed acceptable to obtain an initial solution.

As an illustration, consider a network with $n = 100$ vertices and two target algorithms, X and Y with $T_X = O(n)$ and $T_Y = O(n^2)$, hence, $T_X = O(100)$, $T_Y = O(10,000)$ and $T_X = \sqrt{T_Y}$. Consequently, if one wishes to execute algorithm Y on a network with $n = 10,000$ with a runtime similar to that of algorithm X, the network

Algorithm 4. *Cb*: Contracting of a bipartite network from a matching.

```
Input:
    bipartite network              : $G_l = (V_l, E_l, \sigma_l, \omega_l)$
    matching                       : array of tuples $M = \{(u, v) \mid u, v \in V\}$
Output:
    coarsened bipartite network    : $G_{l+1} = (V_{l+1}, E_{l+1}, \sigma_{l+1}, \omega_{l+1})$

1  $n \leftarrow 1$;
2  sucessor $\leftarrow \emptyset$;
3  $G_{l+1} = (V_{l+1}, E_{l+1}, \sigma_{l+1}, \omega_{l+1}) \leftarrow$ empty bipartite network;
4  forall the $(u, v) \in M$ do
5      sucessor$[u] \leftarrow n$;
6      sucessor$[v] \leftarrow n$;
7      add new super-vertex $sV = \{u, v\}$ to $V_{l+1}$ with id $n$;
8      $\sigma_{l+1}(v) \leftarrow \sigma_l(u) + \sigma_l(v)$;
9      increment $n$;
10 Vertices not merged are inherited by $G_{l+1}$;
11 forall the $(v, u) \in E_l$ do
12     $w \leftarrow$ sucessor$[u]$;
13     $z \leftarrow$ sucessor$[v]$;
14     if $w \neq z$ or they are not super-vertices in $E_{l+1}$ then
15         if $(w, z) \in E_{l+1}$ then
16             $E_{l+1} \leftarrow \omega_{l+1}(w, z) + \omega_l(u, v)$;
17         else
18             $E_{l+1} \leftarrow$ new edge $(w, z)$ with weight $\omega_l(u, v)$;
Return: $G_{l+1} = (V_{l+1}, E_{l+1})$
```

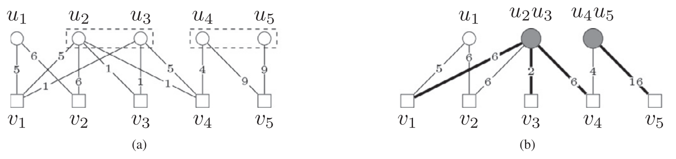


Fig. 5. Contracting of a layer with the GMb_{cn} matching algorithm: (a) selected matching in original bipartite network; (b) contracting process with newly formed super-vertices and modified edges highlighted.

size should be reduced to $\sqrt{n} = 100$, which requires setting $L \approx 7$ and $rf = 0.5$. Of course, the asymptotic complexity of algorithm Y remains quadratic, but the coarsening allows its execution with a runtime equivalent to the linear one.

Analogous to the solution finding phase, the complexity of the uncoarsening phase depends on the target problem (target algorithm), and also on the choice of refinement algorithm, if one is used. In problems such as community detection and classification, it is not necessary to project the solution along the in-

Algorithm 5. *Ub*: Uncoarsening of bipartite networks.

```
Input:
    bipartite network     : $G_{l-1} = (V_{l-1}, E_{l-1}, \sigma_{l-1}, \omega_{l-1})$
    bipartite network     : $G_l = (V_l, E_l, \sigma_l, \omega_l)$
    solution $S_L$        : $S_L = \{P_1, P_2, \dots, P_k\}$
Output:
    projected solution    : $S_{L-1}$

1  new solution $S_{L-1} \leftarrow \emptyset$;
2  forall the $u \in V_{l-1}$ do
3      $w \leftarrow$ successor$[u]$;
4      $P_i \leftarrow$ get_partition$(w)$ with $i \in \{1, \dots, k\}$;
5      assign $u$ to $P_i$;
Return: $S_{L-1}$
```

termediate levels, rather it can be projected directly to the original network, since successors have been stored in an array (or other data structure). In these cases, this phase as implemented in Algorithm 5 would have linear cost in the number of vertices, $O(n)$. As we do not investigate refinement algorithms, an analysis of their asymptotic complexity is suppressed.

5. Experimental results and analysis

We assessed the proposed framework guided by the following research questions:

1. How does coarsening affect a network's topological features at each level?
2. Can the framework improve the efficiency of optimization algorithms without incurring significant losses in the solution quality?
3. How does the framework impact on the scalability of the local search strategy?
4. To which extent can it enable the running of high-cost algorithms on large networks?
5. Is it sufficiently general for handling different combinatorial optimization problems?

We conducted three experimental studies to find answers to the questions. A first study investigated the impact of the coarsening process on the topological properties of a well-known author-paper network. In a second study, the framework was employed in the context of community detection, which is a prototypical application of multilevel strategies. Finally, in a third study, it was employed in dimension reduction in a text classification scenario.

The experiments were executed in a Linux machine with 8-core processor with 3.7 GHz CPU and 64 GB main memory. The framework[1] was implemented in Python with *igraph* library.[2] We report average values obtained from 30 executions for algorithms that rely on random strategies.

5.1. Analysis of topological properties

We considered the scientific collaboration network Cond-Mat,[3] which describes co-authorships of preprints posted from 1995 to 1999 in the Condensed Matter section of the arXiv repository, to address the first research question. It has 38,742 vertices (representing authors and papers) and 58,595 edges (co-authorship relations) (additional information can be found elsewhere [60,61]). Our interest was to observe how a progressive coarsening affects the intrinsic topological properties of a network.

The Cond-Mat network has been extensively analyzed and is known to have characteristic features regarding degree distribution and clustering coefficients, as depicted in Fig. 6. According to Fig. 6(a), the degree distribution follows a power-law relationship characteristic of scale-free networks, with a vast majority of low-degree vertices and a few vertices of very high-degree, i.e., the so-called "hubs".

An inverse relation between vertex clustering coefficients and vertex degrees is also evident in Fig. 6(b). This is a particular feature of this network: the hubs, i.e., authors with many collaborators, have low-clustered neighborhoods and they tend to interact with collaborators from distinct groups, who are not usually collaborators themselves. In contrast, authors with few collaborators have highly-clustered neighborhoods, i.e., they often interact within smaller and more restrict research groups whose members also collaborate with each other.

[1] available from https://www.github.com/alanvalejo/mob.
[2] available from http://www.igraph.org/python/.
[3] available from https://www.toreopsahl.com/datasets/#newman2001.

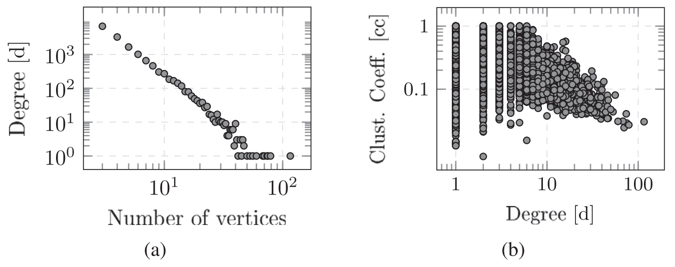


Fig. 6. Scale-free properties of the Cond-Mat bipartite network. (a): histogram of degree distribution; (b): histogram of two-mode local clustering coefficient.

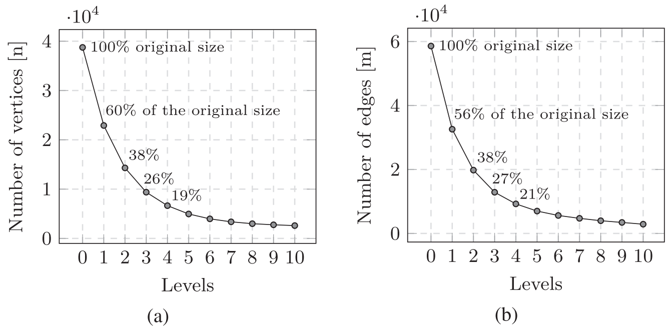


Fig. 7. Percentages of remaining vertices (left) and edges (right) in the networks along the coarsening hierarchy.

We investigated how the scale-free properties of degree distribution and two-mode clustering coefficient are affected as the network is progressively coarsened by matching algorithms *RGMb* and *GMb*, with input parameters set as $L = 10$ and $rf = 0.5$. Fig. 7 shows the reduction factor on $|V| = n$ and $|E| = m$ at each level of the coarsening hierarchy, i.e., starting from the original network (level 0) to the coarsest one (level 10). The curves depict the percentages of remaining vertices (left) and edges (right) relative to the initial numbers.

Figs. 8 and 9 show curves for degree distribution and clustering coefficient, respectively, in each coarsened network, from levels 1 to 10 (the top two rows refer to *RGMb* and the bottom two rows refer to *GMb*). Both algorithms yielded very similar distributions of degree and clustering coefficient, which suggests the random exploration of the solution space adopted by *RGMb* has no strong impact on the topological features of intermediate networks, in comparison with *GMb*.

The characteristic behavior of degree distribution observed in the original network is reasonably preserved in the coarsened models down to level 3, i.e., coarsened networks at levels 1, 2 and 3 still contain few hubs and many low-degree vertices, a behavior that gradually changes from level 4 onwards.

Particularly from level 8, vertex degrees become more homogeneous, until the original topological features have been completely lost in the final network at level 10. A similar pattern is observed in the clustering coefficient, i.e., again, the original behavior of neighborhoods of hub authors is preserved up to level 3. From level 5 onwards, most vertices converge to lower clustering coefficients and the network's characteristic behavior is completely lost from level 8.

At the initial coarsening levels (1, 2 and 3) only vertices with many two-hop common neighbors can compose a matching. The newly formed super-vertices thus preserve the neighborhood properties of their predecessors, hence, the dominant topological properties of the parent network. In other words, it is likely that authors matched at the early coarsening levels indeed have many common collaborators. However, at later levels, authors with few

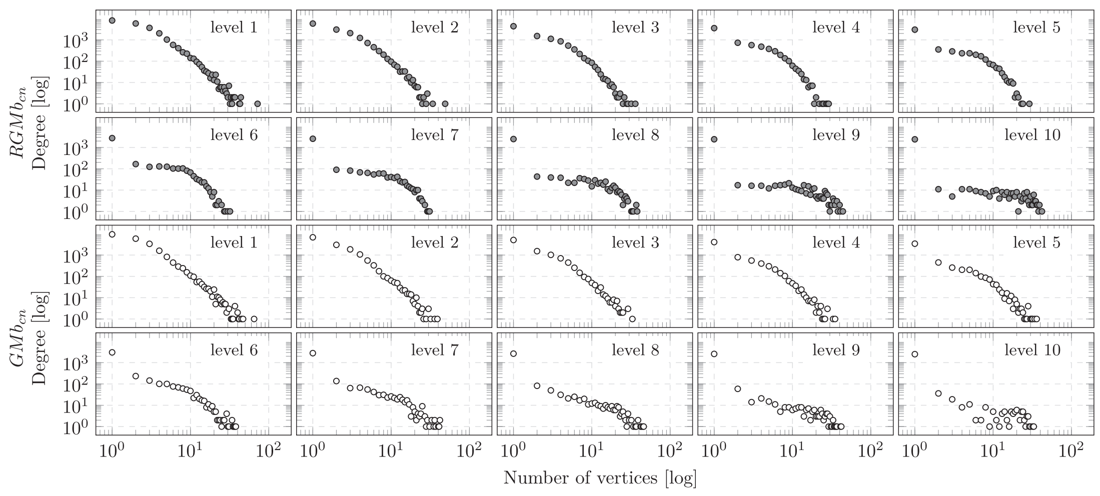


Fig. 8. Degree distribution of the networks at the 10 levels of the coarsening hierarchy ($rf = 0.5$). Top graphs refer to coarsening with *RGMb* (random strategy) and bottom graphs refer to coarsening with *GMb*.

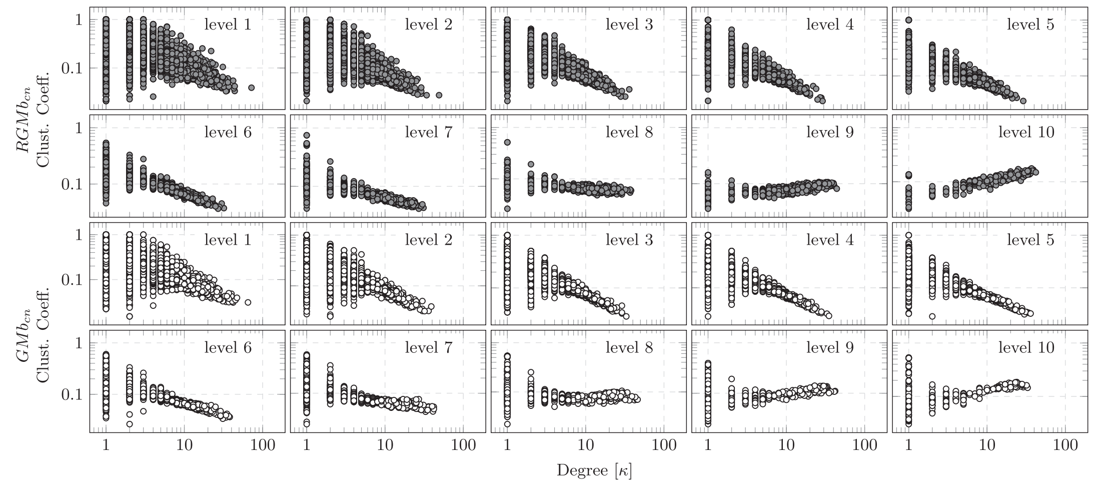


Fig. 9. Two-mode local clustering coefficient of the networks at the 10 levels of the coarsening hierarchy ($rf = 0.5$). Top graphs refer to coarsening with *RGMb* (random strategy) and bottom graphs refer to coarsening with *GMb*.

common collaborators may be forced to join, with corrupts the original topological relations in the network. Notice the sizes of networks at levels 9 and 10 were reduced to nearly 10% of the original network (see Fig. 7), which implies the coarsest networks are mostly formed by heavy-weight super-vertices ($\gg \sigma(sV)$) sparsely connected. Such super-vertices of low-degree form clustered structures with few triangles.

Results from this analysis indicate a limited reduction of an initial network, up to two or three levels, can preserve its relevant topological features. Furthermore, it is evident that choosing the appropriate coarsening level is critical, and depends on properties of the target application and dataset. Establishing a suitable trade-off between accuracy and runtime may require empirical verification in each case. This inherent limitation can be associated to the well-known overfitting/underfitting problem; however, the coarsening phase in our multilevel process "generalizes" the data instead of an objective function. This may explain why, in some cases, better solutions were obtained on the coarsened networks. In general, extensive coarsening reduces execution times of a target algorithm, but it can lead to excessive generalization of the data with significant degradation of topological features, and possibly algorithm accuracy. In contrast, limited coarsening preserves topological features and accuracy, at the expense of higher execution times.

5.2. Performance in benchmark networks for community detection

Research efforts on multilevel optimization have been strongly motivated by community detection (or graph partitioning) problems, which makes this a benchmark problem. Algorithms for community detection in networks split the vertices into disjoint groups (or communities), so as to minimize the number of edges between distinct communities [62]. Barber's modularity optimization [63] is often employed to identify community structures in bipartite networks. Formally, it quantifies the extent of communities formed in both layers relative to a null bipartite network model. Beckett [64] introduced the *LPAwb+* algorithm,[4] which maximizes Barber's modularity through label propagation in weighted bipartite networks and showed it has competitive performance compared with state-of-the-art methods. However, it is a computationally costly algorithm and becomes unfeasible in large-scale networks.

In order to address the second research question, the *MOb* framework was tested considering Beckett's algorithm *LPAwb+* as a target algorithm. Therefore, *MOb* (Algorithm 1) performs the coarsening, runs *LPAwb+* to find the community structure in the

[4] available in https://www.github.com/sjbeckett/weighted-modularity-LPAwbPLUS.

Table 1
MOb applied to target algorithm *LPAwb+* (for the community detection problem): there are two choices of matching algorithm (*RGMb* and *GMb*) employed with two choices of similarity function (S_{cn} and S_{wcn}). All *MOb* algorithms were executed with $rf = 0.5$ and $L = [1, 2, 3]$.

Algorithm	Coarsening		Solution finding	Uncoarsening
	Matching	Contracting		
$MOb - RGMb_{cn}$	*RGMb* with S_{cn}	*Cb*	*LPAwb+*	*Ub*
$MOb - RGMb_{wcn}$	*RGMb* with S_{wcn}			
$MOb - GMb_{cn}$	*GMb* with S_{cn}			
$MOb - GMb_{wcn}$	*GMb* with S_{wcn}			

Table 2
NMI accuracy values of all *MOb* instances and baseline *LPAwb+* in 15 synthetic networks. The values are average results over 30 executions. The highest accuracy values are in bold and values equal to or higher than the baseline (*LPAwb+*) are highlighted (gray background).

Algorithm		Dataset														
Name	Levels [L]	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
LPAwb+	0	0.918	0.926	0.983	0.972	0.964	0.990	0.984	**0.999**	**0.999**	0.985	0.989	**0.996**	0.995	0.987	0.992
$MOb\text{-}RGMb_{cn}$	1	0.984	0.985	0.983	0.988	0.990	0.991	0.992	0.991	0.991	0.992	0.992	0.992	0.991	0.991	0.992
$MOb\text{-}RGMb_{cn}$	2	0.952	0.968	0.963	0.977	0.977	0.976	0.978	0.976	0.978	0.977	0.979	0.981	0.982	0.982	0.980
$MOb\text{-}RGMb_{cn}$	3	0.866	0.910	0.923	0.936	0.943	0.944	0.952	0.954	0.950	0.957	0.954	0.957	0.956	0.955	0.957
$MOb\text{-}RGMb_{wcn}$	1	0.982	0.988	0.986	0.988	0.990	0.990	0.991	0.987	0.992	0.992	0.991	0.991	0.992	0.992	0.992
$MOb\text{-}RGMb_{wcn}$	2	0.942	0.960	0.966	0.968	0.974	0.973	0.978	0.978	0.977	0.978	0.977	0.977	0.979	0.979	0.978
$MOb\text{-}RGMb_{wcn}$	3	0.905	0.922	0.948	0.954	0.954	0.951	0.953	0.951	0.960	0.959	0.958	0.961	0.959	0.961	0.960
$MOb\text{-}GMb_{cn}$	1	**0.994**	**0.993**	**0.993**	**0.995**	**0.996**	**0.996**	**0.995**	0.993	0.994	**0.995**	**0.995**	0.995	**0.996**	**0.995**	**0.996**
$MOb\text{-}GMb_{cn}$	2	0.981	0.986	0.982	0.983	0.988	0.987	0.988	0.987	0.988	0.988	0.988	0.988	0.988	0.990	0.990
$MOb\text{-}GMb_{cn}$	3	0.901	0.934	0.963	0.966	0.971	0.968	0.973	0.974	0.974	0.975	0.972	0.975	0.976	0.975	0.977
$MOb\text{-}GMb_{wcn}$	1	0.990	0.989	0.992	0.992	0.995	0.993	0.992	0.992	0.993	0.994	**0.995**	0.995	0.995	0.994	0.994
$MOb\text{-}GMb_{wcn}$	2	0.969	0.985	0.985	0.979	0.988	0.985	0.987	0.985	0.987	0.987	0.989	0.988	0.988	0.989	0.989
$MOb\text{-}GMb_{wcn}$	3	0.965	0.973	0.973	0.972	0.975	0.979	0.977	0.977	0.978	0.977	0.978	0.980	0.978	0.979	0.979

coarsest network, and projects the solution to obtain the community structure in the original network. Our goal was to investigate whether *MOb* can yield solutions statistically equivalent in quality to the standard *LPAwb+*, whilst increasing its scalability. Results are compared with those obtained with the standard *LPAwb+*, used as baseline.

We investigated the performance of the four instances of *MOb*, listed in Table 1. They were executed with parameter settings $rf = 0.5$ and $L = [1, 2, 3]$ in a set of 15 synthetic weighted bipartite networks, identified as R1-R15 (hereafter each *MOb* instance is referred to by its name).

Synthetic networks were obtained with the community model described in [64], which creates unbalanced and randomly positioned community structures. Networks of sizes $n = |V_1 + V_2|$ were generated within the range [1, 000; 15, 000] at increments of 1,000 and the number of communities was set to $0.01{*}n$. Edge weights were randomly assigned from a skewed negative binomial distribution and noise was introduced in the connection patterns by reconnecting a percentage of the edges between and within communities.

Performance was measured in terms of accuracy, by means of the NMI (normalized mutual information), which compares the solution found by a selected algorithm with the baseline [65], we also measured execution times. Table 2 shows the NMI accuracy values in the 15 networks. The highest values are in bold and values equal to or higher than baseline *LPAwb+* are highlighted with a gray background. The best performances were achieved by $MOb - GMb_{cn}$ with one level of coarsening ($L = 1$) on 11 out of the 15 networks. Baseline *LPAwb+* yielded the best performance in three networks, whereas $MOb - RGMb_{cn}$ with $L = 3$ yielded the worst results.

Limited coarsening levels (mainly $L = 1$) yielded higher accuracy values, reinforcing that a controlled coarsening can filter the solution space by joining promising vertex pairs and removing irrelevant high-cost solutions, while preserving important topological features. In contrast, accuracy deteriorated with more extensive coarsening ($L = 3$), which does not necessarily preserve the original topological properties and is likely to blur the boundaries between adjacent communities. The effect of parameter L depends on network size, i.e.; differences in algorithm accuracy are likely to decrease as the network sizes increase, hinting that higher values of L might be successfully adopted in handling larger networks.

Although *LPAwb+* achieved the best performance in three networks out of the fifteen, the corresponding accuracy values attained by the *MOb* instances are very close for these networks. Indeed, in these specific cases accuracy values differ up to 0.006 in R8, up to 0.005 in R9 and up to 0.001 in R12. Interestingly, all *MOb* instances yielded similar accuracy values and more stable results than the standard *LPAwb+*. The accuracy values obtained with $MOb - GMb_{cn}$ and $L = 1$, for instance, are within the range [0.994, 0.996], whereas for the standard *LPAwb+* they are within the range [0.918, 0.999], as shown in Figs. 10 and 11.

Fig. 10 depicts the averages and standard deviations of the accuracy values, whereas Fig. 11 shows the dispersion of their distribution and outliers, considering in both cases the alternative settings of parameter L. The bar plots in Fig. 10(a) reveal superior performance and stability of the four *MOb* instances when $L = 1$, confirmed by their higher average accuracies and narrower standard deviations. When $L = 2$, only $MOb - GMb_{cn}$ and $MOb - GMb_{wcn}$ yielded a slightly superior solution in comparison to *LPAwb+*. For $L = 3$, *LPAwb+* yielded better results than any *MOb* instance, a consequence of the extensive network reduction.

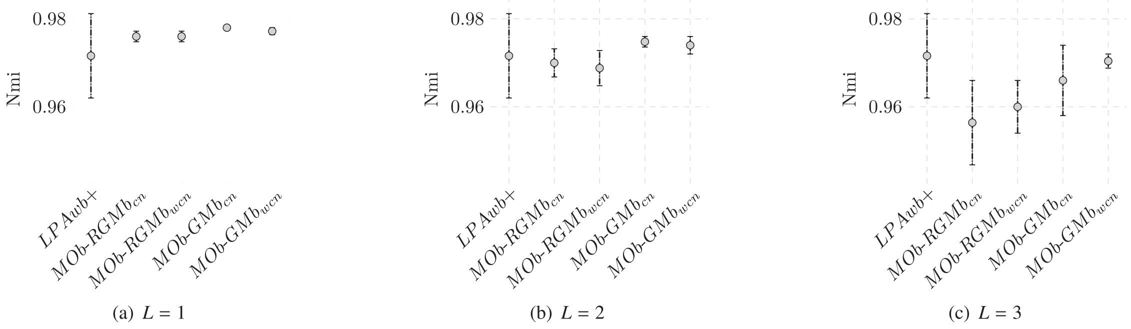


Fig. 10. Averages and standard deviations of the NMI accuracy values obtained with *LPAwb+* and four *MOb* instances in three settings of parameter *L* (number of levels). (a) $L = 1$, (b) $L = 2$ and (c) $L = 3$.

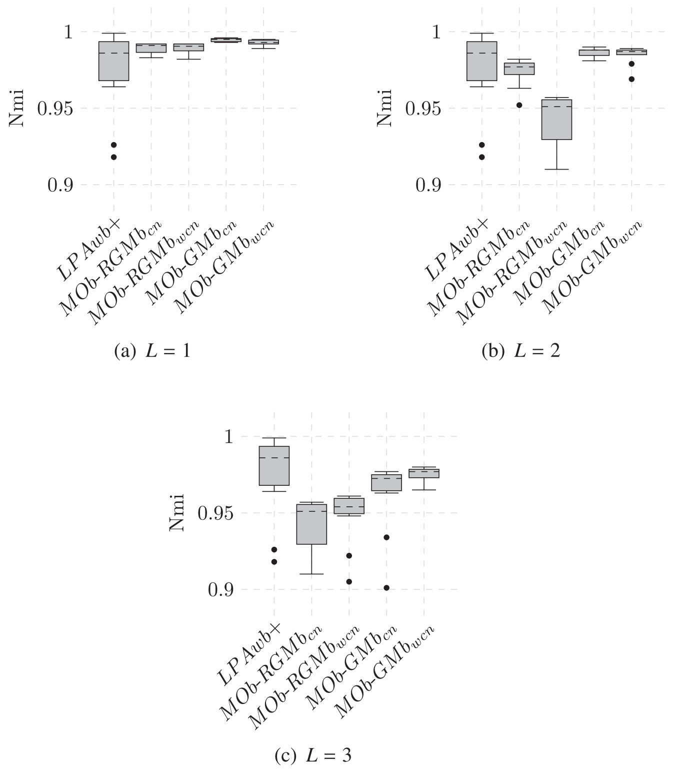


Fig. 11. Shape distribution, variability, and median of the accuracy values yielded by *LPAwb+* and the four instances of *MOb* considering three settings of parameter *L*. (a) $L = 1$, (b) $L = 2$ and (c) $L = 3$.

The box plots in Fig. 11(a) reveal that for $L = 1$ all *MOb* instances yielded accuracy values within a narrower distribution and higher averages than *LPAwb+*.

We can conclude *MOb* instances yielded, in general, higher accuracy and improved stability in comparison to *LPAwb+*. In summary, the experimental evidence regarding solution quality (average, standard deviation and dispersion of the accuracy values) suggests the multilevel framework stabilizes and improves the performance of the algorithm.

A Nemenyi post-hoc test [66] was applied to the results in Table 2 to verify statistical differences in the algorithms performances and the results are shown in Fig. 12 for (a) $L = 1$, (b) $L = 2$ and (c) $L = 3$. The critical difference (CD) is indicated at the top of each diagram and the algorithms' average ranks are placed on the horizontal axes, with the best ranked algorithms to the left. A black line connects algorithms if no significant difference has been detected among them. According to the Nemenyi statistics, the critical value for comparing the mean-ranking of two different algorithms at 95 percentile is 1.58.

Let us consider the outcome of the post-hoc test for $L = 1$, i.e., when the number of vertices *n* is reduced by a factor of two, shown in Fig. 12(a). $MOb-GMB_{cn}$ was ranked best, followed by $MOb-GMb_{wcn}$ and $MOb-RGMb_{wcn}$ and then *LPAwb+*. Furthermore, $MOb-GMB_{cn}$, $MOb-GMb_{wcn}$ and $MOb-RGMb_{wcn}$ presented statistically significant differences compared with standard *LPAwb+*. Interestingly, for $L = 2$ (Fig. 12(b)), $MOb-GMB_{cn}$ and $MOb-GMb_{wcn}$ remain ranked first, however, no statistically significant difference was observed in relation to *LPAwb+*. Finally, for $L = 3$ (Fig. 12(b)), *LPAwb+* was ranked first, with no statistically significant difference observed in relation to $MOb-GMb_{cn}$ or $MOb-GMb_{wcn}$. However, parameter settings $L = 3$ and $rf = 0.5$ implied, in this case, in reducing the original size by a factor of 75%, which explains the poor performance of all *MOb* instances.

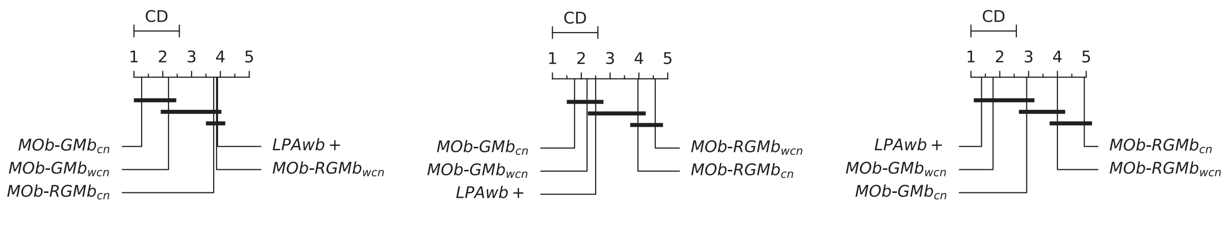
We also assessed the scalability of the *MOb* instances to investigate the third and fourth research questions. Their performance was analyzed considering each individual network and the total time spent in the experiments. Table 3 shows the absolute execution times (in seconds) on each network - values refer to the average times from 30 executions of each scenario.

The longest execution time of standard *LPAwb+* was 302,442 s (time to process the largest network) and the shortest was 14 s (time to process the smallest one). $MOb-GMb_{wcn}$ was the most expensive *MOb* instance, consuming ($L = 1$) 25,674 s on the largest network and 3 s on the smallest one. Therefore, regarding its maximum and minimum execution times, respectively, $MOb-GMb_{wcn}$ run 11.8 to 4.6 times faster than the standard *LAPwb+*. The maximum and minimum running times of the least expensive *MOb* instance, $MOb-RGMb_{cn}$ ($L = 3$), were 1,310 s and 1 s, respectively. Therefore, $MOb-RGMb_{cn}$ run 230 to 14 times faster than *LAPwb+*.

The analysis in Section 5.1 revealed that a network coarsened at level 3 has roughly 25% of its original size. Let us consider, for example, network R15 ($n = 15,000$): we know it has been reduced to 3750 vertices at level 3 and algorithm $MOb-RGMb_{cn}$ processed it in 1,310 s (see Table 3). This is close to the execution time of standard *LPAwb+* on network *R4* (of size $n = 4,000$, similar to the size of R15 coarsened at level 3), i.e., 904 s. As $MOb-RGMb_{cn}$ executes the coarsening/uncoarsening steps, the actual time spent running *LPAwb+* to find the solution is roughly similar in both cases, but $MOb-RGMb_{cn}$ is handling a network nearly four times larger.

The total time spent running the experiments was 1,267.307 s, or nearly 352 h. Fig. 13 shows the contribution of each algorithm to the total time, considering both absolute values (seconds) (a) and relative values (percentages) (b).

In the best case, the *MOb* instances reduced execution time from 891,875 s (nearly 208.8 h) required by the standard *LPAwb+* to 4,552.4 s (1.26 h), which implies *LPAwb+* was nearly 195 times

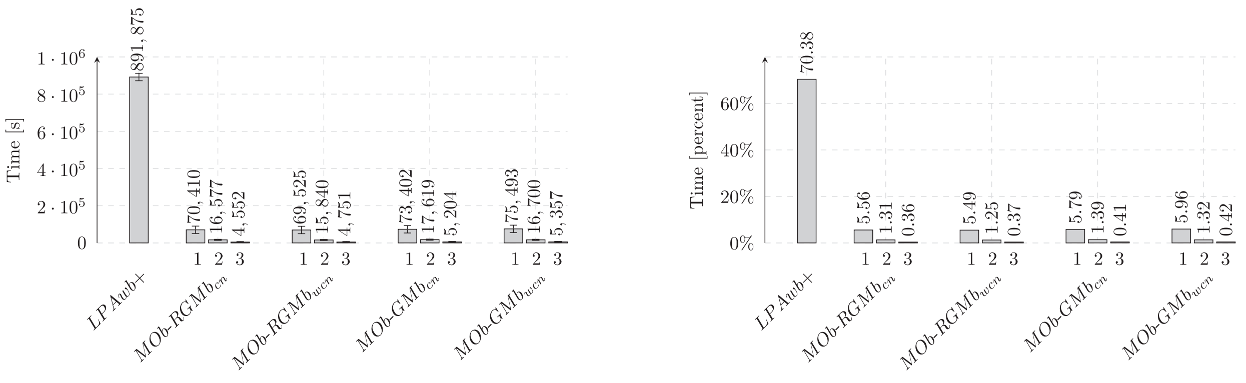


(a) $L = 1$ (b) $L = 2$ (c) $L = 3$

Fig. 12. Nemenyi post-hoc test for *LPAwb*+ and the four *MOb* instances.

Table 3
Absolute runtime (seconds) of *LPAwb*+ and four *MOb* instances on each network.

Algorithm		Dataset															sum
Name	Levels [L]	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	
LPAwb+	0	14	96	308	904	2782	2800	7146	15,925	39,197	56,119	66,729	75,990	97,392	224,032	302,442	891,875
MOb − RGMb$_{cn}$	1	2	10	31	77	160	328	616	1106	1787	4148	5386	8108	10,242	15,235	23,174	70,410
MOb − RGMb$_{cn}$	2	1	3	8	19	42	77	154	267	428	1081	1296	1989	2372	3559	5283	16,577
MOb − RGMb$_{cn}$	3	1	1	4	7	14	25	49	86	142	304	389	586	694	940	1310	4552
MOb − RGMb$_{wcn}$	1	3	11	34	79	174	322	643	1424	1715	4192	5335	8582	9355	15,121	22,535	69,525
MOb − RGMb$_{wcn}$	2	1	4	9	24	44	101	165	282	460	1049	1399	1911	2245	3281	4866	15,840
MOb − RGMb$_{wcn}$	3	1	3	6	11	18	33	59	91	139	323	396	522	713	1056	1381	4751
MOb − GMb$_{cn}$	1	2	13	33	78	165	329	623	1111	2680	4145	5561	6705	9091	18,136	24,732	73,402
MOb − GMb$_{cn}$	2	1	5	11	24	46	84	168	298	721	1088	1492	1657	2455	4013	5555	17,619
MOb − GMb$_{cn}$	3	1	3	6	13	22	34	62	90	239	332	471	481	672	1180	1596	5204
MOb − GMb$_{wcn}$	1	3	13	39	83	180	338	688	1135	2864	4392	5819	6317	8932	19,016	25,674	75,493
MOb − GMb$_{wcn}$	2	3	8	15	32	55	96	187	326	744	1100	1447	1719	2345	4050	4574	16,700
MOb − GMb$_{wcn}$	3	3	7	12	20	31	45	80	117	287	392	469	536	745	1233	1380	5357
sum		36	178	514	1371	3733	4613	10,639	22,257	51,403	78,664	96,189	115,104	147,253	310,853	424,502	1,267,307



(a) Runtime (seconds) (b) Runtime (percentage)

Fig. 13. Contribution of each algorithm to the total time of the experiments: absolute values (seconds) (a) and percentages (b), 100% runtime corresponds to 1.127.300 seconds. Each bar refers to a *MOb* instance executed with a particular setting of parameter $L = [1, 2, 3]$, with the specific execution time and standard deviations shown on top in (a) and relative percent times shown on top in (b).

slower than its *MOb* instantiations. Executing *LPAwb*+ consumed over 70% of the time spent in the experiments, whereas roughly 6% of the time was spent running the *MOb* instances.

Finally, we assessed the impact of each phase (coarsening, target algorithm on coarsest network and uncoarsening) on the execution time of the multilevel process; we analyzed algorithm behavior separately on each network and then in relation to the total time of the experiments.

The relative contributions of each multilevel phase for each network are shown in Fig. 14 (for legibility, we show bars for 12 out of the 15 networks). On the smallest network ($n = 1,000$) the coarsening phase consumed nearly 52% of the total execution time and the local search step consumed nearly 45%. On the other hand, as the networks increase, the time spent on the coarsening relative to the solution finding gradually decreases. On the larger networks, the coarsening phase consumed roughly 1% of the total execution time, in contrast to roughly 99% of the solution finding phase. The time spent on the uncoarsening phase was negligible.

Fig. 15 shows the relative contribution of each multilevel phase to the total time of the experiments. In general, the coarsening phase consumed less than 1% of the total execution time, the solution finding phase (executing *LPAwb*+) consumed over 99%, and the time spent in the uncoarsening phase was negligible. These results indicate coarsening and uncoarsening exerted no significant influence on the scalability of the multilevel process and provide empirical evidence the multilevel strategy is a promising approach to scale network optimization algorithms.

From this empirical investigation we conclude: (i) the proposed *MOb* approach yielded more accurate and stable results compared to the standard *LPAwb*+; (ii) although solution quality degrades as the network is progressively coarsened, runtime drops drastically at each additional coarsening level; hence, a successful solution re-

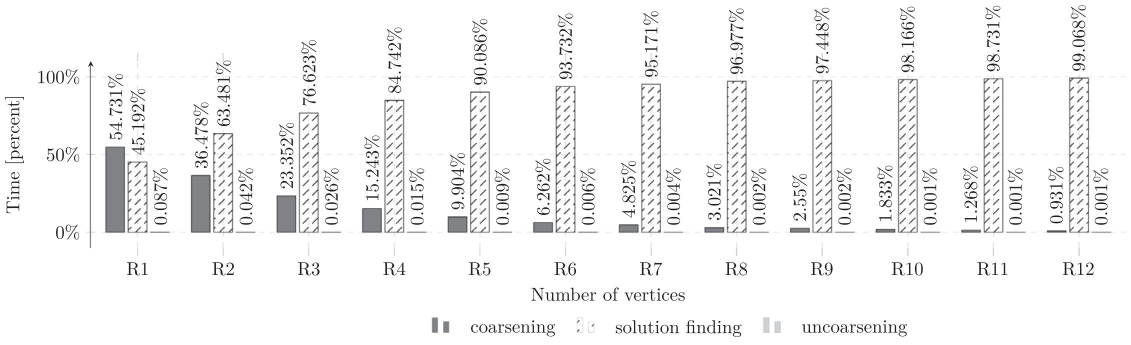


Fig. 14. Relative contribution (percentage) of each multilevel phase to the execution time, on different network sizes n (from 1000 for R1 to 12,000 for R12).

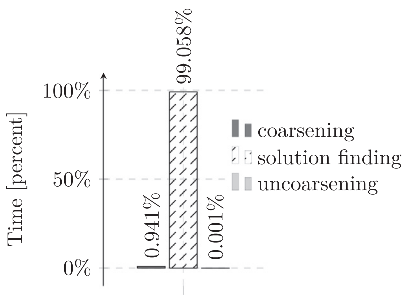


Fig. 15. Average relative contribution (percentage of execution time) of each stage of MOb in all 15 networks.

quires establishing a suitable trade-off between accuracy and execution time.

5.3. A test case on dimensionality reduction

We illustrate how *MOb* framework can be adapted to perform dimensionality reduction in the context of text classification, having the *k*-Nearest Neighbor classifier (*kNN*) as the target algorithm, in order to exemplify its application in a different kind of optimization problem.

Documents are often represented, in text classification tasks, as multidimensional feature vectors, in which each dimension maps a particular term. As the dimensionality of the representation space has strong impact in classification performance, such tasks are often preceded by a dimension reduction step. Specifically, a *kNN* classifier that employs a naÏve search strategy has time complexity $O(ndk)$ for a fixed k, where n is the cardinality of the training set and d denotes the dimensionality of the document representation.

Alternatively, a document corpus can be represented as a bipartite network $G = ((V_1 \cup V_2), E, \sigma, \omega)$, where $V_1 = \{d_1, \ldots, d_r\}$ is the set of documents and $V_2 = \{t_1, \ldots, t_s\}$ is the set of terms. An edge $(u,\ v)$ exists if term t_u occurs in document d_v, and the term's frequency determines the corresponding edge weight $w(u,\ v)$. Such a network is represented as a bi-adjacency matrix A_{rxs}, where $r = |V_1|$, $s = |V_2|$ and $A_{u,v} = w(u, v)$ if $(u,\ v) \in E$. Dimensionality reduction is aimed at obtaining a lower dimensional matrix $A'_{r'\text{x}s'}$, with $r' = r$ and $s' << s$.

The proposed solution is described in Algorithm 6, ***m**ultilevel **d**imensionality **r**eduction* (Mdr), which only requires a coarsening phase (lines 1–3). It takes as inputs the initial bipartite network G, the term layer t_l to be coarsened, the desired maximum number of levels L and reduction factor *rf*, and returns the bi-adjacency matrix of the coarsest network (line 4).

We adopted $RGMb_{cn}$ as the contracting algorithm in the *Mdr* implementation, and used *Mdr* in combination with a *kNN* classifier, assuming $k = 3$ and the Euclidean distance as similarity measure, hereafter referred to as $Mdr - kNN$. The results obtained considering a reduction factor $rf = 0.5$ and L varying in the range [1,2,3,4,5,6,7,8,9,10] were compared to an equivalent *kNN* setting that employed *PCA* (Principal Components Analysis) [67] for dimensionality reduction ($PCA - kNN$). The complexity of *PCA* is $O(d^2n + d^3)$, where n denotes the number of samples and d the data dimensionality.

Implementations $Mdr - kNN$ and $PCA - kNN$ were used to classify thirteen real document-term networks available from the literature, described in Table 4, considering a cross-validation with permutation testing and ten-fold cross-validation to estimate classification error. The training set was randomly split into ten equal-sized subsets, so the classification model was trained in nine subsets and tested on the remaining one.

Fig. 16 shows the performance of the $Mdr - kNN$ classifier as a function of the number of levels in the coarsening hierarchy, where $L = 0$ refers to classification with no dimensionality reduction. Fig. 16(a)–(c) show accuracy, dimensionality of the representation (number of terms) and execution times, respectively.

A moderate decrease in accuracy values is observed up to level $L = 5$. In general, accuracy is stable at the early coarsening levels, which suggests the first coarsening iterations have limited impact in solution quality, as it is more likely that highly correlated terms are being matched. Fig. 16(a) shows accuracies up to level 5. Moreover, since from level 5 onwards the coarsening yields no significant reduction in the number of terms it may be interrupted at this point. Also, a sharp reduction in execution times is observed only up to this level. We remind the analysis presented in Section 5.1, which showed a network coarsened at level 5 has roughly 15% its original size.

We compared the classification accuracies of *kNN* without dimension reduction and $Mdr - kNN$ (with coarsening at $L = [1, 3, 5]$) and those of $PCA - kNN$ (with equivalent dimensionality reduction). The results are shown in Table 5. The highest accuracy values are shown in bold and values equivalent or superior to baseline *kNN* are shown with a gray background. $Mdr - kNN$ achieved the best performance in five out of the thirteen networks; $PCA - kNN$ performed best in seven networks and standard *kNN* performed best in only one of them.

A Nemenyi post-hoc test was applied to the results in Table 5 in order to detect statistical differences among the algorithms. Demsar post-hoc test requires each algorithm and dataset to be independent, therefore, we perform the dimensionality reduction at each level separately. The results are shown in Fig. 17 for (a) $L = 1$, (b) $L = 2$ and (c) $L = 3$. According to Nemenyi statistics, in all diagrams, the critical value for comparing the mean-ranking of two different algorithms at 95 percentile is 0.92. No significant difference was observed between the algorithms, therefore, they are connected by a bold line in each diagram. Albeit differences are not significant, Fig. 17(a) shows $Mdr - kNN$ was ranked best for $L = 1$,

Algorithm 6. Mdr: multilevel dimensionality reduction.

```
Input:
    bipartite network                              : $G = (V, E, \sigma, \omega)$
    term layer                                     : $t_l \in \{1, 2\}$
    maximal number of levels for term layer        : $L \in [0, n] \subset \mathbb{Z}$
    reduction factor for term layer                : $rf \in (0, 0.5] \subset \mathbb{R}$
Output:
    low-dimensional bi-adjacency matrix            : A'
1  while $l \leq L$ or layer $l$ is as small as desired do
2      $M \leftarrow$ matching($G_l$, $t_l$, rf);
3      $G_{l+1} \leftarrow$ contracting($G_l$, M);
4  matrix A' $\leftarrow$ bi-adjacency matrix of $G_l$;
Return: A'
```

Table 4
Properties of text collections considered.

	cstr	oh0	oh5	0h10	syskillwebert	opinosis	classic4	hitech	industry	irish	la1s	enron	acm
Documents	299	1003	918	1050	334	6457	7095	2301	8817	1660	3204	13,199	3493
Terms	1726	3183	3013	3239	4340	2693	7749	10,000	10,000	8659	10,000	10,000	10,000
Domain	Scientific	Medical	Medical	Medical	Web pages	Sentiment	Scientific	News	Web page	Sentiment	News	E-mail	Scientific

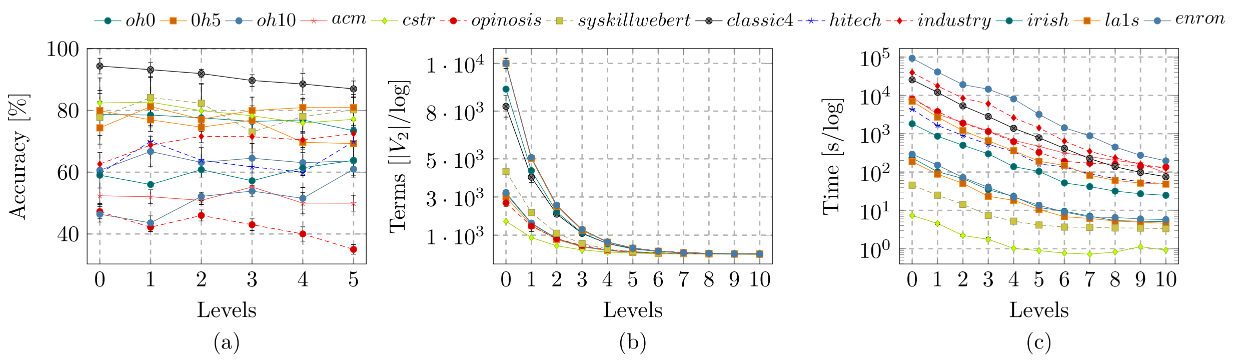
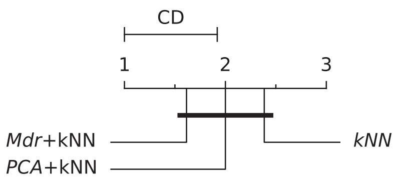


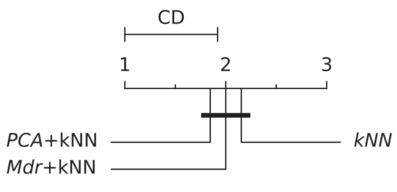
Fig. 16. Performance (accuracy values) of the *Mdr* – *kNN* classifier. *L* = 0 corresponds to classification with no dimension reduction applied.

Table 5
Accuracy values obtained with standard *kNN*, *Mdr* – *kNN* and *PCA* – *kNN*. The highest accuracy values are shown in bold and values equivalent or superior to the baseline *kNN* are highlighted in gray.

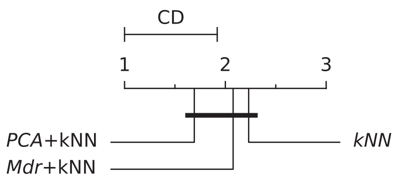
Dataset	kNN	L = 1		L = 3		L = 5	
		MdrkNN	PCA-kNN	Mdr-kNN	PCA-kNN	Mdr-kNN	PCA-kNN
cstr	82.44 ± 8.09	82.71 ± 8.04	75.86 ± 7.15	80.10 ± 6.10	79.31 ± 2.02	78.30 ± 3.20	**86.21** ± 7.10
oh0	78.72 ± 6.84	81.15 ± 3.31	**85.00** ± 3.98	78.95 ± 2.02	76.00 ± 4.31	75.90 ± 3.32	76.00 ± 6.20
oh5	80.27 ± 5.09	78.83 ± 3.73	76.92 ± 4.31	76.88 ± 4.17	**81.32** ± 3.11	69.27 ± 3.89	74.36 ± 5.35
oh10	60.62 ± 5.75	66.92 ± 4.95	**75.24** ± 4.21	64.54 ± 5.04	72.38 ± 3.29	64.38 ± 2.16	73.33 ± 4.23
opinosis	47.25 ± 2.32	42.18 ± 1.42	43.57 ± 1.20	45.90 ± 4.92	**55.19** ± 8.15	35.98 ± 3.06	49.15 ± 5.78
syskillwebert	77.80 ± 8.70	**87.91** ± 6.57	87.88 ± 5.14	75.91 ± 3.65	75.06 ± 6.45	80.23 ± 2.12	78.00 ± 5.65
acm	52.37 ± 2.54	52.96 ± 2.26	54.44 ± 2.62	**55.23** ± 3.15	53.00 ± 2.98	49.98 ± 1.87	49.10 ± 4.87
hitech	59.57 ± 3.55	**70.00** ± 2.59	64.35 ± 3.24	61.74 ± 2.34	66.10 ± 3.10	69.90 ± 2.55	66.90 ± 3.50
industry	62.66 ± 5.60	71.62 ± 3.05	66.86 ± 1.98	71.50 ± 6.12	**78.30** ± 4.12	72.89 ± 3.34	75.20 ± 4.32
irish	59.04 ± 4.28	63.86 ± 2.34	57.83 ± 4.32	58.23 ± 3.87	56.90 ± 4.07	**63.91** ± 5.12	63.10 ± 5.98
la1s	74.38 ± 6.74	81.25 ± 4.10	**83.75** ± 5.20	80.00 ± 4.12	82.12 ± 2.23	80.94 ± 4.80	82.30 ± 4.76
classic4	**94.39** ± 6.04	93.82 ± 2.55	93.51 ± 3.15	89.72 ± 4.01	75.87 ± 2.68	87.00 ± 3.12	91.90 ± 3.28
enron	46.40 ± 5.18	52.16 ± 5.09	47.76 ± 2.98	53.89 ± 2.90	55.00 ± 3.77	**61.00** ± 2.14	60.10 ± 6.12



(a) $L = 1$



(b) $L = 3$



(c) $L = 5$

Fig. 17. Nemenyi post-hoc test for *kNN* and its variant with *Mdr* and *PCA* for dimensionality reduction in three settings of parameter L. (a) $L = 1$, (b) $L = 3$ and (c) $L = 5$.

whereas for $L = 2$ and $L = 3$ $PCA - kNN$ was ranked best (Fig. 17(b) and (c)). Therefore, $Mdr - kNN$ proved competitive in terms of accuracy, in comparison to $PCA - kNN$ and baseline *kNN*.

This case study has been presented as a preliminary investigation on the feasibility of extending the proposed multilevel framework to other combinatorial problems beyond community detection. The *Mdr* algorithm deserves further consideration and could incorporate additional capabilities, e.g., it would be convenient to reduce the feature space to a target dimensionality, rather than by a given reduction factor. Furthermore, the coarsening algorithm could take into account intrinsic characteristics of specific kinds of document-term networks by means of customized matching algorithms. Moreover, *Mdr* can be employed in connection with other classification algorithms.

6. The application of *MOb* to other combinatorial optimization problems

It is relatively straightforward to apply *MOb* to several combinatorial optimization problems beyond community detection. For instance, in Section 5.3 we illustrated its application to handle a dimensionality reduction problem over a bipartite network in which the two vertex layers represent objects and features, respectively. In this context, only the feature layer was coarsened and the reduced feature space is given directly by the adjacency matrix of the coarsest network. Likewise, it can be easily instantiated to handle overlapping or fuzzy community detection or classification problems.

In overlapping community detection, for each decomposed $sV \in V_{l+1}$, its original vertices $\{u, v\} \in V_l$ are assigned to the same set of communities as the corresponding super-vertex. If the coarsened network has a fuzzy structure, the strength of a vertex' pertinence to a community will be equal to that of its super-vertex. Similarly, in a classification problem original vertices $\{u, v\} \in V_i$ should be assigned to the same class of their super-vertex. Therefore, instantiating the framework to handle either problem would require just minor modifications in the projection algorithm (Algorithm 5).

MOb might also be useful to support interactive visualization of large-scale bipartite networks, by means of navigation over a hierarchy of coarsened networks, which would demand a data structure to keep these intermediate networks.

Instantiation to other scenarios is not necessarily as straightforward, and may require further modifications in the proposed algorithms. For example, in the edge clustering problem (also called link communities) the contracting algorithm used in coarsening phase would require adjustments. Whereas in community detection a vertex inherits the same group assignment of its super-vertex, here each edge must inherit the connections from its super-edge. Therefore, an edge $e \in E_{l+1}$ incident to $s_V \in V_{l+1}$ must refer to the edges incident to vertices $\{u, v\} \in V_l$. Algorithm 4 does not implement this function; however, it could be done by keeping an additional data structure similar to the `successor vector`.

Another possible application is in link prediction or recommendation problems, albeit this poses a more complex scenario for generalization. Link prediction methods rely on similarity between vertices, since similar vertices are likely to share common links. However, such information is not explicitly given for the super-vertices in a coarsened bipartite network. Nonetheless, the framework could be employed to reduce the number of required operations, i.e. super-vertices might be created grouping vertices with shared h-hop neighbors, thus filtering the search space. Link prediction could be performed in the uncoarsening phase through the decomposed super-vertices, and the solution finding and uncoarsening phases would be executed simultaneously.

As a final consideration, the matching algorithm used for coarsening should be carefully designed to incorporate the specific characteristics of each problem and context.

7. Conclusion and further research

Inspired by the potential of general-purpose multilevel strategies to scale optimization algorithms we have introduced algorithms of a novel multilevel optimization framework (*MOb*) for bipartite networks, and illustrated its application on two combinatorial optimization problems. Our framework accounts for the specificities of bipartite networks and provides a powerful tool for handling a variety of problems.

We investigated three empirical scenarios to illustrate strengths and limitations of the proposed *MOb* framework. A first study has shown that a controlled coarsening preserves relevant topological features of a network. A second study described an application in community detection, showing that *MOb* combined with a proper local search strategy can drastically improve speedup of a classic community detection algorithm while preserving solution quality. Finally, in a third study we considered text classification to illustrate how the general framework can be instantiated to handle different combinatorial optimization problems.

Our results provide compelling evidence that *MOb* offers a competitive approach to scale existing methods while preserving solution quality, and reinforce its usefulness for handling combinatorial optimization problems in large bipartite networks. Furthermore, the framework is flexible and can be adjusted to incorporate alternative and novel coarsening methods targeted at specific applications. We also discuss some general guidelines for future applications of combinatorial problems, such as link prediction, edge clustering, and interactive graph visualization.

Identifying the level of coarsening that will yield a suitable trade-off between accuracy and execution times is a critical issue in applying the proposed multilevel strategy. Currently, this is done by means of empirical investigation in each application problem and dataset, but it certainly deserves further investigation.

We also plan as future work to extend the framework to handle problems defined in heterogeneous networks, where edges connect vertices of multiple types. It would be applicable, e.g., to document-word networks indicating associations of the type document-word, word-word, and document-document; or networks describing relations between words, documents and authors. We are also interested in investigating distributed or parallel paradigms, as well as in application of *MOb* to supervised and unsupervised classification tasks. An implementation of the general framework is currently available and can be downloaded from https://www.github.com/alanvalejo/mob.

Acknowledgments

Author A. Valejo is supported by a scholarship from the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES) grants PROEX-6513393/D. This work has been partially supported by the State of São Paulo Research Foundation (FAPESP) grants 15/14228-9 and 17/05838-3; and the Brazilian Federal Research Council (CNPq) grants 302645/2015-2 and 301847/2017-7.

References

[1] R.G. Rossi, A. de Andrade Lopes, S.O. Rezende, Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts, Inf. Process. Manage. 52 (2) (2016) 217–257.
[2] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, A.L. Barabasi, The large-scale organization of metabolic networks, Nature 407 (6804) (2000) 651–654.
[3] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.
[4] Z. Wang, D. Zhang, X. Zhou, D. Yang, Z. Yu, Z. Yu, Discovering and profiling overlapping communities in location-based social networks, IEEE Trans. Syst. Man, Cybern. 44 (4) (2014) 499–509.
[5] M.E.J. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. U. S. A. (PNAS) 98 (2) (2001) 404–409.
[6] A.S. Asratian, T.M.J. Denley, R. Häggkvist, Bipartite Graphs and their Applications, Cambridge University Press, New York, NY, USA, 1998.
[7] G. Zhang, J. Lu, Y. Gao, Multi-Level Decision Making: Models, Methods and Applications, Springer-Verlag Berlin, Heidelberg, 2015.
[8] J. Han, J. Lu, Y. Hu, G. Zhang, Tri-level decision-making with multiple followers: model, algorithm and case study, Inf. Sci. 311 (2015) 182–204.
[9] J. Lu, J. Han, Y. Hu, G. Zhang, Multilevel decision-making: a survey, Inf. Sci. 346347 (2016) 463–487.
[10] A. Brandt, Multilevel computations: review and recent developments, in: Proceedings of the Multigrid Methods: theory, Applications and Supercomputing, 110, 1988, pp. 35–62.
[11] G. Karypis, V. Kumar, Analysis of multilevel graph partitioning, in: Proceedings of the ACM/IEEE Conference on Supercomputing, 1995, pp. 1–19.
[12] S.-H. Teng, Algorithms for Parallel Processing, Springer New York, pp. 247–276.
[13] A. Noack, R. Rotta, Multi-level algorithms for modularity clustering, in: J. Vahrenhold (Ed.), Proceedings of the Experimental Algorithms: 8th International Symposium SEA, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 257–268.
[14] A. Valejo, J. Valverde-Rebaza, B. Drury, A.A. Lopes, Multilevel refinement based on neighborhood similarity, in: Proceedings of the International Database Engineering & Applications Symposium (IDEAS), in: IDEAS '14, ACM, New York, NY, USA, 2014, pp. 67–76.
[15] A. Valejo, J. Valverde-Rebaza, A.A. Lopes, A multilevel approach for overlapping community detection, in: Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS), 2014, pp. 390–395.
[16] D. Lasalle, G. Karypis, Multi-threaded modularity based graph clustering using the multilevel paradigm, J. Parallel Distrib. Comput. 76 (2015) 66–80.
[17] C. Walshaw, Multilevel refinement for combinatorial optimisation problems, Ann. Oper. Res. 131 (1) (2004) 325–372.
[18] C. Walshaw, Hybrid Metaheuristics: An Emerging Approach to Optimization, vol. 114, Springer Berlin Heidelberg, pp. 261–289.
[19] C. Walshaw, A multilevel algorithm for force-directed graph drawing, in: Proceedings of the Graph Drawing: 8th International Symposium, GD, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 171–182.
[20] C. Walshaw, A multilevel approach to the travelling salesman problem, Ann. Oper. Res. 50 (5) (2002) 862–877.
[21] C.-E. Bichot, Graph Partitioning, ISTE - Wiley, pp. 27–63.
[22] M. Chernoskutov, Y. Ineichen, C. Bekas, Heuristic algorithm for approximation betweenness centrality using graph coarsening, Procedia Comput. Sci. 66 (2015) 83–92.
[23] L. Lü, T. Zhou, Link prediction in weighted networks : the role of weak ties, Europhys. Lett. (EPL) 89 (1) (2010) 18001.
[24] M.E.J. Newman, Mixing patterns in networks, Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys. 67 (2) (2003) 026126.
[25] M. Latapy, C. Magnien, N.D. Vecchio, Basic notions for the analysis of large two-mode networks, Soc. Networks 30 (1) (2008) 31–48.
[26] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. (SISC) 20 (1) (1998) 359–392.
[27] S.T. Barnard, H.D. Simon, Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems, Concurrency 6 (2) (1994) 101–117.
[28] C. Walshaw, M. Cross, Mesh partitioning: a multilevel balancing and refinement algorithm, SIAM J. Sci. Comput. (SISC) 22 (1) (1998) 63–80.
[29] P. Korosec, J. Silc, B. Robic, Mesh partitioning: a multilevel ant-colony-optimization algorithm, in: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), 00, 2003, pp. 1–8.
[30] A. Abou-Rjeili, G. Karypis, Multilevel algorithms for partitioning power-law graphs, in: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), 2006. 124–124
[31] S. Oliveira, S.C. Seok, A multilevel approach to identify functional modules in a yeast protein-protein interaction network, in: Proceedings of the Computational Science (ICCS): 6th International Conference, Reading, 2006, pp. 726–733.
[32] S.T. Barnard, Pmrsb: parallel multilevel recursive spectral bisection, in: Proceedings of the ACM/IEEE conference on Supercomputing (CDROM), 1995, pp. 1–27.
[33] G. Karypis, V. Kumar, Parallel multilevel graph partitioning, in: Proceedings of the International Conference on Parallel Processing, 1996, pp. 314–319.
[34] G. Karypis, V. Kumar, A parallel algorithm for multilevel graph partitioning and sparse matrix ordering, J. Parallel Distrib. Comput. 48 (1) (1998) 71–95.
[35] G. Karypis, V. Kumar, Parallel multilevel k-way partitioning scheme for irregular graphs, in: Proceedings of the ACM/IEEE Conference on Supercomputing, 1996.
[36] C. Walshaw, M. Cross, Parallel optimisation algorithms for multilevel mesh partitioning, Parallel Comput. 26 (i) (2000) 1635–1660.
[37] K. Schloegel, G. Karypis, V. Kumar, Parallel multilevel algorithms for multi-constraint graph partitioning, in: Proceedings of the Euro-Par Parallel Processing, Proceedings, 1900, 2000, pp. 296–310.
[38] R. Baños, C. Gil, J. Ortega, F.G. Montoya, A parallel multilevel metaheuristic for graph partitioning, J. Heuristics 10 (3) (2004) 315–336.
[39] A. Trifunovic, W.J. Knottenbelt, Parkway 2.0: a parallel multilevel hypergraph partitioning tool, in: Proceedings of the Computer and Information Sciences (ISCIS), 2004, pp. 789–800.
[40] A. Trifunovic, W.J. Knottenbelt, A parallel algorithm for multilevel k-way hypergraph partitioning, in: Proceedings of the Parallel and Distributed Computing, IEEE, 2004, pp. 114–121.
[41] K. Erciye, A. Alp, G. Marshall, Serial and parallel multilevel graph partitioning using fixed centers, in: Proceedings of the Current Trends in Theory and Practice of Computer Science, 2005, pp. 127–136.
[42] E.A. Schweitz, D.P. Agrawal, A parallelization domain oriented multilevel graph partitioner, IEEE Trans. Comput. 51 (12) (2007) 1435–1441.
[43] C. Walshaw, M. Cross, Mesh Partitioning Techniques and Domain Decomposition Methods, Saxe-Coburg Publications, pp. 27–58.
[44] D. Lasalle, G. Karypis, Multi-threaded graph partitioning, in: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), 2013, pp. 225–236.
[45] M.E.J. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. U. S. A. (PNAS) 103 (23) (2006) 8577–8582.
[46] H.N. Djidjev, A scalable multilevel algorithm for graph clustering and community structure detection, in: Proceedings of the Algorithms and Models for the Web-Graph: Fourth International Workshop, 2008, pp. 117–128.
[47] P. Schuetz, A. Caflisch, Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement, Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys. 77 (4) (2008) 1–7.
[48] Z. Ye, S. Hu, J. Yu, Adaptive clustering algorithm for community detection in complex networks, Phys. Rev. E 78 (2008) 046115.
[49] R. Rotta, A. Noack, Multilevel local search algorithms for modularity clustering, J. Exp. Algorithmics 16 (2) (2011) 2.1.
[50] H.N. Djidjev, M. Onus, Scalable and accurate graph clustering and community structure detection, IEEE Trans. Parallel Distrib. Syst. 24 (5) (2013) 1022–1029.
[51] C. Walshaw, A Multilevel Approach to the Graph Colouring Problem, Technical Report, School of Computing and Mathematical Science, Univeristy of Greenwich, London, UK, 2001.
[52] I.O. Oduntan, A multilevel search algorithm for feature selection in biomedical data, Dept. Computer Science, Ph.D. thesis. University of manitoba, 2005.
[53] C. Dai, P.C. Li, M. Toulouse, A cooperative multilevel tabu search algorithm for the covering design problem, in: Proceedings of the International Conference on Artificial Evolution, 3871 LNCS, Springer Berlin Heidelberg, 2006, pp. 119–130.
[54] C. Blum, M.Y. Valles, Multi-level ant colony optimization for dna sequencing by hybridization, in: Proceedings of the International Workshop on Hybrid Metaheuristics, Springer, 2006, pp. 94–109.
[55] D. Rodney, A. Soper, C. Walshaw, The application of multilevel refinement to the vehicle routing problem, in: Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling (CISsched), IEEE, 2007, pp. 212–219.
[56] C. Zhang, F. Wang, A multilevel approach for learning from labeled and unlabeled data on graphs, Pattern Recognit. 43 (6) (2010) 2301–2314.
[57] M. Karpinski, W. Rytter, Fast Parallel Algorithms for Graph Matching Problems, Oxford University Press, 1998.
[58] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, S. Tao, Neighborhood based fast graph search in large networks, in: Proceedings of the International Conference on Management of Data (SIGMOD), in: SIGMOD '11, ACM, New York, NY, USA, 2011, pp. 901–912.
[59] G. Kollias, M. Sathe, O. Schenk, A. Grama, Fast parallel algorithms for graph similarity and matching, J. Parallel Distrib. Comput. 74 (5) (2014) 2400–2410.
[60] M.E.J. Newman, Scientific collaboration networks: i. network construction and fundamental results, Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys. 64 (1) (2001) 1–8.
[61] M.E.J. Newman, Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality, Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys. 64 (1) (2001) 016132.

[62] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3–5) (2010) 75–174.
[63] M.J. Barber, Modularity and community detection in bipartite networks, Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys. 76 (6) (2007) 1–11.
[64] S.J. Beckett, Improved community detection in weighted bipartite networks, R. Soc. Open Sci. 3 (1) (2016) 140536.
[65] V. Labatut, Generalized measures for the evaluation of community detection methods, Int. J. Soc. Netw. Anal. Min. (SNAM) 2 (1) (2015) 44–63.
[66] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[67] I. Jolliffe, Principal Component Analysis, Springer Series in Statistics, Springer, 2002.