

## MIT Open Access Articles

*Partition-Merge: Distributed Inference and Modularity Optimization*

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

**Citation:** Blondel, Vincent, Jung, Kyomin, Kohli, Pushmeet, Shah, Devavrat and Won, Seungpil. 2021. "Partition-Merge: Distributed Inference and Modularity Optimization." IEEE Access, 9.

**As Published:** 10.1109/ACCESS.2021.3070490

**Publisher:** Institute of Electrical and Electronics Engineers (IEEE)

**Persistent URL:** <https://hdl.handle.net/1721.1/143875>

**Version:** Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

**Terms of use:** Creative Commons Attribution 4.0 International license



Received February 5, 2021, accepted March 22, 2021, date of publication April 1, 2021, date of current version April 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3070490

# Partition-Merge: Distributed Inference and Modularity Optimization

VINCENT BLONDEL<sup>1</sup>, KYOMIN JUNG<sup>2,5</sup>, (Member, IEEE), PUSHMEET KOHLI<sup>3</sup>,  
DEVAVRAT SHAH<sup>1,4</sup>, (Senior Member, IEEE), AND SEUNGPIIL WON<sup>1,2,5</sup>

<sup>1</sup>Applied Mathematics, Université catholique de Louvain, 1348 Louvain-la-Neuve, Belgium

<sup>2</sup>Department of Electrical and Computer Engineering, Seoul National University, Gwanak-gu 08826, South Korea

<sup>3</sup>DeepMind, London NIC 4AG, U.K.

<sup>4</sup>MIT, Cambridge, MA 1239, USA

<sup>5</sup>Automation and Systems Research Institute, Seoul National University, Seoul 08826, South Korea

Corresponding author: Kyomin Jung (kjung@snu.ac.kr)


This work was supported in part by the Army Research Office through MURI under Award 58153-MA-MUR, in part by ARO MURI under Grant 133668-5079809, in part by NSF under Grant CMMI-1462158, in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology under Grant 2012032786, in part by the Brain Korea 21 (BK21) FOUR Program of the Education and Research Program for Future Information and Communication Technology (ICT) Pioneers, Seoul National University, in 2021, and in part by the Automation and Systems Research Institute (ASRI), Seoul National University.

**ABSTRACT** This paper presents a novel meta-algorithm, Partition-Merge (PM), which takes existing centralized algorithms for graph computation and makes them distributed and faster. In a nutshell, PM divides the graph into small subgraphs using our novel randomized partitioning scheme, runs the centralized algorithm on each partition separately, and then *stitches* the resulting solutions to produce a global solution. We demonstrate the efficiency of the PM algorithm on two popular problems: computation of Maximum A Posteriori (MAP) assignment in an arbitrary pairwise Markov Random Field (MRF) and modularity optimization for community detection. We show that the resulting distributed algorithms for these problems become fast, which run in time linear in the number of nodes in the graph. Furthermore, PM leads to performance comparable – or even better – to that of the centralized algorithms as long as the graph has polynomial growth property. More precisely, if the centralized algorithm is a  $C$ -factor approximation with constant  $C \geq 1$ , the resulting distributed algorithm is a  $(C + \delta)$ -factor approximation for any small  $\delta > 0$ ; and even if the centralized algorithm is a non-constant (e.g., logarithmic) factor approximation, then the resulting distributed algorithm becomes a constant factor approximation. For general graphs, we compute explicit bounds on the loss of performance of the resulting distributed algorithm with respect to the centralized algorithm. To show the efficiency of our algorithm, we conducted extensive experiments both on real-world networks and on synthetic networks. The experiments demonstrate that the PM algorithm provides a good trade-off between accuracy and running time.

**INDEX TERMS** Approximate MAP, graphical model, modularity optimization, partition.

## I. INTRODUCTION

Graphical representation for data has become central to modern large-scale data processing applications. For many of these applications, large-scale data computation boils down to solving problems defined over massive graphs. While the theory of centralized algorithms for graph problems is getting reasonably well developed, their distributed (as well as parallel) counterparts are still poorly understood [2]–[5].

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Da Lin .

These studies remain very active areas of the current investigation, along with graph partitioning [6]–[9].

### A. SUMMARY OF RESULTS

In this paper, we take an important step towards this challenge. Specifically, we present a meta-algorithm, Partition-Merge (PM), that makes existing centralized (exact or approximate) algorithms for graph computation distributed and faster without loss of performance, and in some cases, even improving performance. The PM meta-algorithm is based on our novel partitioning of the graph into small disjoint subgraphs. In a nutshell, PM partitions the graph

into small subgraphs, runs the centralized algorithm on each partition separately (which can be done in a distributed or parallel manner), and finally *stitches* the resulting solutions to produce a global solution. We apply the PM algorithm to two representative classes of problems: the MAP computation in a pairwise MRF and modularity optimization-based graph clustering.

The paper establishes that for any graph that satisfies the polynomial growth property, the resulting distributed PM-based implementation of the original centralized algorithm is a  $(\mathcal{C} + \delta)$ -approximation algorithm whenever the centralized algorithm is a  $\mathcal{C}$ -approximation algorithm for some constant  $\mathcal{C} \geq 1$ . In this expression,  $\delta$  is a small number that depends on a tunable parameter of the algorithm that affects the size of the induced subgraphs in the partition; the larger the subgraph size, the smaller the  $\delta$ . More generally, if the centralized algorithm is an  $\alpha(n)$ -approximation (with  $\alpha(n) = o(n)$ ) for a graph of size  $n$ , the resulting distributed algorithm becomes a constant factor approximation for graphs with geometric structure! The computational complexity of the algorithm scales linearly in  $n$ . Thus, our meta-algorithm can make centralized algorithms faster, distributed and improve their performance.

The algorithm applies to any graph structure, but strong guarantees on performance, as stated above, require geometric structure.<sup>1</sup> However, it is indeed possible to explicitly evaluate the loss of performance induced by the distributed implementation compared to the centralized algorithm, as stated in Section IV-B.

A cautionary remark is in order. Indeed, by no means, this algorithm means to answer all problems in distributed computation. Specifically, for dense graphs, this algorithm is likely to exhibit poor performances, and definitely such graph structure would require a very different approach. Our meta-algorithm requires that the underlying graph problem is *decomposable* or *Markovian* in a sense. Not all problems have this structure, and therefore these problems require a different way to think about them.

To verify the validity of the PM algorithm, we have applied it both on real-world networks and on synthetic graphs of various sizes, comparing it with the original centralized algorithms. For MAP inference, we used the sequential tree-reweighted max-product message passing (TRW-S) [10] with the energy function defined in Section V-B. For modularity optimization, we selected three different algorithms from the old-fashioned way to state-of-the-art: Girvan-Newman (GN) [11], Clauset-Newman-Moore (CNM) [12], and Louvain-Method (LM) [13]. Overall, as long as the network is decomposable like grid graphs, PM considerably reduces the running time while maintaining the performance of the original centralized algorithm. In the case of a comparatively dense network, such

<sup>1</sup>Roughly speaking, a graph is said to have geometric structures or polynomial growth property when the number of nodes within distance  $r$  of any given node grows no faster than a polynomial function of  $r$ .

as the Barabási-Albert model, PM produces a decent result, although the efficiency of PM is relatively low.

In our experiments, PM particularly performs better when applied on well-distributed regular networks and when the centralized algorithm has high complexity. Moreover, the performance with respect to MAP inference on graphs with a high average degree is outstanding; PM achieves similar performance to the centralized algorithm in less than half the time. Besides, we researched what value of partition radius offers better efficiency for our algorithm. We used a fixed partition radius to understand the performance of PM according to its value, and it leads to the conclusion that PM generally operates to its best efficiency when a partition radius is close to the average distance between a pair of nodes in the network.

## B. RELATED WORK AND OUR CONTRIBUTIONS

The results of this paper, on the one hand, are related to a long list of works on designing distributed algorithms for decomposable problems. On the other hand, the applications of our method to MAP inference in pairwise MRFs and clustering relate our work to a large number of results in these two respective problems. We will only be able to discuss very closely related work here.

We start with the most closely related work on the use of graph partitioning for distributed algorithm design. Such an approach is quite old; see, e.g., [14], [15] and [16] for a detailed account of the approach until 2000. More recently, such decompositions have found a wide variety of applications, including local-property testing [17]. All such decompositions are useful for *homogeneous* problems, e.g., finding maximum-size matching or independent set rather than the *heterogeneous* maximum-weight variants of it. To overcome this limitation, Jung and Shah [18] introduced a different (somewhat stronger) notion of decomposition, built upon [15], for minor-excluded graphs. All of these results effectively partition the graph into small subgraphs and then solve the problem inside each small subgraph using exact (dynamic programming) algorithms. While this results in a  $(1 + \epsilon)$ -approximation algorithm for any  $\epsilon > 0$  with computation scaling essentially linearly in the graph size ( $n$ ), the computation constant depends super-exponentially on  $1/\epsilon$ . Therefore, even with  $\epsilon = 0.1$ , the algorithms become unmanageable in practice.

As the main contribution of this paper, we first propose a novel graph decomposition scheme for graphs with geometry or polynomial growth structure. Then we establish that utilizing this decomposition scheme along with *any* centralized algorithm (instead of dynamic programming) for solving the problem inside the partition leads to performance comparable (or better) to that of the centralized algorithm for a graph with polynomial growth. Unlike the dynamic programming approach, the resulting distributed algorithm becomes very fast in practice if the centralized algorithm inside the partition runs fast. We verify the effectiveness of the PM algorithm through experiments, finding that this decomposition scheme

actually produces a similar performance (better in some cases) to that of the centralized algorithm in a very short time. As mentioned above, the result is established for both MAP in pair-wise MRF and modularity optimization-based clustering. Similar guarantees can be obtained for minor-excluded graphs as well using the scheme utilized in [18].

In this work, we focus our attention on two questions, as mentioned above. However, the method suggested here can be applied broadly to generic “optimization” when (i) the constraints are represented through graph structure over variable nodes, (ii) there is a notion of “default” assignment to variables that satisfies all the constraints. Indeed, problems such as the maximum weight independent set, vertex cover, or MAP inference in generic pair-wise Markov Random Field are instances of this. And these are instances of NP-complete problems.

### C. MAP INFERENCE

Computing the exact Maximum a Posteriori (MAP) solution in a general probabilistic model is an NP-hard problem. Several algorithmic approaches have been developed to obtain approximate solutions for these problems. Most of these methods work by making “local updates” to the assignment of the variables. Starting from an initial solution, the algorithms search the space of all possible local changes that can be made to the current solution (also called move space) and choose the best amongst them.

One such algorithm (which has been rediscovered multiple times) is called Iterated Conditional Modes, or ICM for short. Its local update involves selecting (randomly or deterministically) a variable of the problem. ICM assigns a value to the selected variable keeping all other variables fixed, which results in a solution with the maximum probability. This process is repeated by selecting other variables until the probability cannot be increased further. The local step of the algorithm can be seen as performing inference in the smallest decomposed subgraph possible.

Another family of methods is related to max-product belief propagation (cf. [19] and [20]). In recent years, a sequence of results suggests an intimate relationship between the max-product algorithm and a natural linear programming relaxation – for example, see [21]–[25]. Many of these methods can be seen as making local updates to partitions of the dual problem [26], [27].

We also note that the Swendsen-Wang algorithm (SW) [28], a local flipping algorithm, has a philosophy similar to ours in that it repeats a process of randomly partitioning the graph and computing an assignment. However, the graph partitioning of SW is fundamentally different from ours, and there is no known guarantee for the error bound of SW.

In summary, all the approaches thus far with provable guarantees for the local update-based algorithm are primarily for linear or, more generally, convex optimization setup.

### D. MODULARITY OPTIMIZATION FOR CLUSTERING

The notion of modularity optimization was introduced by Newmann [29] to identify the communities or clusters in a

network structure. Since then, it has become quite popular as a metric to find communities or clusters in various networked data cf. [13], [30], [31]. The major challenge has been designing an approximation algorithm for modularity optimization (which is computationally hard in general) that can operate in a distributed manner and provide performance guarantees. Such algorithms with provable performance guarantees are known only for few cases, notably logarithmic approximation of [32] via a centralized solution.

Our contribution in the context of modularity optimization lies in showing that indeed it is a *decomposable* problem and therefore admits a distributed and fast approximation algorithm through our approach.

### E. ORGANIZATION

The rest of the paper is organized as follows. Section II describes the problem statement and preliminaries. Section III describes our main algorithms, and Section IV presents analyses of our algorithms. The proofs of our main theorems are given in the Appendix (Section VII-A and Section VII-B). Section V and Section VI present the setup and results of an experiment, respectively, and Section VII presents the conclusion.

## II. SETUP

### A. GRAPHS

Our interest is in processing networked data represented through an undirected graph  $G = (V, E)$  with  $n = |V|$  vertices and  $E$  being the edge set. Let  $m = |E|$  be the number of edges. Graphs can be classified structurally in many different ways: trees, planar, minor-excluded, geometric, expanding, and so on. We shall establish results for graphs with geometric structure or polynomial growth, which we define next. A graph  $G = (V, E)$  induces a natural “graph metric” on vertices  $V$ , denoted by  $\mathbf{d}_G : V \times V \rightarrow \mathbb{R}_+$  with  $\mathbf{d}_G(i, j)$  given by the length of the shortest path between  $i$  and  $j$ ; defined as  $\infty$  if there is no path between them.

*Definition 1 (Graph With Polynomial Growth):* We say that a graph  $G$  (or a collection of graphs) has polynomial growth of degree (or growth rate)  $\rho$ , if for any  $i \in V$  and  $r \in \mathbb{N}$ ,

$$|\mathbf{B}_G(i, r)| \leq C \cdot r^\rho,$$

where  $C > 0$  is a universal constant and  $\mathbf{B}_G(i, r) = \{j \in V \mid \mathbf{d}_G(i, j) < r\}$ .

Note that interesting values of  $C, \rho$  are integral between  $\{0, 1, \dots, n\}$ , and it is easy to compute in  $O(mn)$  time. Therefore we will assume knowledge of  $C, \rho$  for algorithm design. A large class of graph model naturally fall into the graphs with polynomial growth. To begin with, the standard  $d$ -dimensional regular grid graphs have polynomial growth rate  $d$ . More generally, in recent years, in the context of computational geometry and metric embedding, the graphs with finite doubling dimensions have become a popular object of study [33]. It can be checked that a graph with doubling dimension  $\rho$  is also a graph with polynomial growth rate  $\rho$ .

Finally, the popular geometric graph model, where nodes are placed arbitrarily in some Euclidean space with some minimum distance separation, and two nodes have an edge between them if they are within a certain finite distance, has a finite polynomial growth rate [34].

### B. PAIR-WISE GRAPHICAL MODEL AND MAP

For a pair-wise Markov Random Field (MRF) model defined on a graph  $G = (V, E)$ , each vertex  $i \in V$  is associated with a random variable  $X_i$  which we shall assume to be taking value from a finite alphabet  $\Sigma$ ; the edge  $(i, j) \in E$  represents a form of “dependence” between  $X_i$  and  $X_j$ . More precisely, the joint distribution is given by

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) \propto \prod_{i \in V} \phi_i(x_i) \cdot \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \quad (1)$$

where  $\phi_i : \Sigma \rightarrow \mathbb{R}_+$  and  $\psi_{ij} : \Sigma^2 \rightarrow \mathbb{R}_+$  are called node and edge potential functions.<sup>2</sup> The question of interest is to find the maximum a posteriori (MAP) assignment  $\mathbf{x}^* \in \Sigma^n$ , i.e.

$$\mathbf{x}^* \in \arg \max_{\mathbf{x} \in \Sigma^n} \mathbb{P}[\mathbf{X} = \mathbf{x}].$$

Equivalently, from the optimization point of view, we wish to find an optimal assignment of the problem

$$\begin{aligned} & \text{maximize } \mathcal{H}(\mathbf{x}) \quad \text{over } \mathbf{x} \in \Sigma^n, \quad \text{where} \\ & \mathcal{H}(\mathbf{x}) = \sum_{i \in V} \ln \phi_i(x_i) + \sum_{(i,j) \in E} \ln \psi_{ij}(x_i, x_j). \end{aligned}$$

For completeness and simplicity of exposition, we assume that the function  $\mathcal{H}$  is finitely valued over  $\Sigma^n$ . However, the results of this paper extend for hard constrained problems such as the *hardcore* or *independent set* model. We call an algorithm  $\alpha$  approximation for  $\alpha \geq 1$  if it always produces assignment  $\hat{\mathbf{x}}$  such that

$$\frac{1}{\alpha} \mathcal{H}(\mathbf{x}^*) \leq \mathcal{H}(\hat{\mathbf{x}}) \leq \mathcal{H}(\mathbf{x}^*).$$

### C. SOCIAL DATA AND CLUSTERING/COMMUNITY DETECTION

Alternatively, in a social setting, vertices of graph  $G$  can represent individuals, and edges represent some form of interaction between them. For example, consider a cultural show organized by students at a university with various acts. Let there be  $n$  students in total who have participated in one or more acts. Place an edge between two students if they participated in at least one act together. Then the resulting graph represents the interaction between students in terms of acting together.

Based on this observed network, the goal is to identify the set of all acts performed and its “core” participants. The true answer, which classifies each student/node into the acts in which s/he performed, would lead to partitions of nodes in which a node may belong to multiple partitions. Our interest is in identifying disjoint partitions, which would, in this

example, roughly mean identification of “core” members of acts.

In general, it is not clear what is the appropriate criteria to select a disjoint partition of  $V$  given  $G$ . Newman [29] proposed the notion of modularity as a criterion. The intuition behind it is that a cluster or community should be as distinct as possible from being “random”. The modularity of a partition of nodes is defined as the fraction of the edges that fall within the disjoint partitions minus the expected such fraction if edges were distributed at random with the same node degree sequences. Formally, the modularity of a subset  $S \subset V$  is defined as

$$M(S) = \sum_{i,j \in S} \left( A_{ij} - \frac{d_i d_j}{2m} \right), \quad (2)$$

where  $A_{ij} = 1$  iff  $(i, j) \in E$  and 0 otherwise,  $d_i = |\{k \in V : (i, k) \in E\}|$  is the degree of node  $i \in V$ , and  $m = |E|$  represents the total number of edges in  $G$ . More generally, the modularity of a partition of  $V$ ,  $V = S_1 \cup \dots \cup S_\ell$  for some  $1 \leq \ell \leq n$  with  $S_i \cap S_j = \emptyset$  for  $i \neq j$ , is given by

$$\mathcal{M}(S_1, \dots, S_\ell) = \frac{1}{2m} \left( \sum_{i=1}^{\ell} M(S_i) \right). \quad (3)$$

The modularity optimization approach [29] proposes to identify the community structure as the disjoint partitions of  $V$  that maximizes the total modularity, defined as per (3), among all possible disjoint partitions of  $V$  with ties broken arbitrarily. The resulting clustering of nodes is the desired answer.

We shall think of clustering as assigning *colors* to nodes. Specifically, given a coloring  $\chi : V \rightarrow \{1, \dots, n\}$ , two nodes  $i$  and  $j$  are part of the same cluster (partition) iff  $\chi(i) = \chi(j)$ . With this notation, any clustering of  $V$  can be represented by some such coloring  $\chi$  and vice versa. Therefore, modularity optimization is equivalent to finding a coloring  $\chi$  such that its modularity  $\mathcal{M}(\chi)$  is maximized, where

$$\mathcal{M}(\chi) = \frac{1}{2m} \sum_{i,j \in V} \mathbf{1}_{\{\chi(i)=\chi(j)\}} \left( A_{ij} - \frac{d_i d_j}{2m} \right).$$

Here  $\mathbf{1}_{\{\cdot\}}$  is the indicator function with  $\mathbf{1}_{\{\text{true}\}} = 1$  and  $\mathbf{1}_{\{\text{false}\}} = 0$ . Let  $\chi^*$  be a clustering that maximizes the modularity. Then, as before, an algorithm will be said  $\alpha$ -approximate if it produces  $\hat{\chi}$  such that

$$\frac{1}{\alpha} \mathcal{M}(\chi^*) \leq \mathcal{M}(\hat{\chi}) \leq \mathcal{M}(\chi^*). \quad (4)$$

### III. PARTITION-MERGE ALGORITHM

We describe a parametric meta-algorithm for solving the MAP inference and modularity optimization. The meta-algorithm uses two parameters; a large constant  $K \geq 1$  and a small real number  $\varepsilon \in (0, 1)$  to produce a partition of  $V = V_1 \cup \dots \cup V_p$  so that each partition  $V_j$ ,  $1 \leq j \leq p$  is small. We will specify the values of  $K$  and  $\varepsilon$  in Section IV. The meta-algorithm uses an existing centralized algorithm to

<sup>2</sup>For simplicity of the analysis we assume strict positivity of  $\phi_i$ 's and  $\psi_{ij}$ 's.

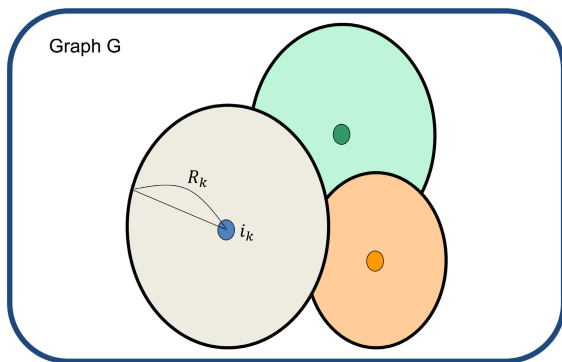


solve the original problem on each of these partitioned sub-graphs  $G_j = (V_j, E_j)$  independently where  $E_j = (V_j \times V_j) \cap E$ . The resulting assignment leads to a candidate solution for the problem on the entire graph. As we establish in Section IV, this becomes a pretty good solution. Next, we describe the algorithm in detail.

*Step 1 (Partition):* We wish to create a partition of  $V = V_1 \cup \dots \cup V_p$  for some  $p$  with  $V_i \cap V_j = \emptyset$  for  $i \neq j$  so that the number of edges crossing partitions are small. The algorithm for such partitioning is iterative. Initially, no node is part of any partition. Order the  $n$  nodes arbitrarily, say  $i_1, \dots, i_n$ . In iteration  $k \leq n$ , choose node  $i_k$  as the pivot. If  $i_k$  belongs to  $\cup_{\ell=1}^{k-1} V_\ell$ , then set  $V_k = \emptyset$ , and move to the next iteration if  $k < n$  or else the algorithm concludes. If  $i_k \notin \cup_{\ell=1}^{k-1} V_\ell$ , choose a radius  $R_k \leq K$  at random with distribution

$$\mathbb{P}(R_k = \ell) = \begin{cases} \varepsilon(1 - \varepsilon)^{\ell-1} & \text{for } 1 \leq \ell < K \\ (1 - \varepsilon)^{K-1} & \text{for } \ell = K. \end{cases} \quad (5)$$

Let  $V_k$  be the set of all nodes in  $V$  that are within distance  $R_k$  of  $i_k$ , but that are not part of  $V_1 \cup \dots \cup V_{k-1}$ . Since we execute this step only if  $i_k \notin \cup_{\ell=1}^{k-1} V_\ell$  and  $R_k \geq 1$ ,  $V_k$  will be non-empty. At the end of the  $n$  iterations, we have a partition of  $V$  with at most  $n$  non-empty partitions. Let the non-empty partitions of  $V$  be denoted as  $V = V_1 \cup \dots \cup V_p$  for some  $p \leq n$ . A caricature of an iteration is described in Figure 1.



$$\Pr[R_k = i] = \varepsilon(1 - \varepsilon)^{i-1} \text{ for } i=1,2,3,\dots,(K-1)$$

FIGURE 1. A pictorial description of an iteration of the graph partitioning.

*Step 2 [Merge (Solving the Problem)]:* Given the partition  $V = V_1 \cup \dots \cup V_p$ , consider the graphs  $G_k = (V_k, E_k)$  with  $E_k = (V_k \times V_k) \cap E$  for  $1 \leq k \leq p$ . We shall apply a centralized algorithm for each of these graph  $G_1, \dots, G_k$  separately. Specifically, let  $\mathcal{A}$  be an algorithm for MAP or clustering: the algorithm may be exact (e.g., one solving a problem by exhaustive search over all possible options, or dynamic programming), or it may be an approximation algorithm (e.g.,  $\alpha$ -approximate for any graph). We apply  $\mathcal{A}$  for each subgraph separately.

- o For MAP inference, this results in an assignment to all variables since in each partition each node is assigned some value and collectively all nodes are covered by the partition. Declare resulting global assignment, say  $\hat{\mathbf{x}}$  as the solution for MAP.

- o For modularity optimization, nodes in each partition  $V_j$  are clustered. We declare the union of all such clusters across partitions as the global clustering. Thus two nodes in different partitions are always in different clusters; two nodes in the same partition are in different clusters if the centralized algorithm applied to that partition clusters them differently.

*Computation Cost:* The computation cost of the partitioning scheme scales linearly in the number of edges in the graph. The computation cost of solving the problem in each of the components  $G_1, \dots, G_p$  depends on component sizes and on how the computation cost of algorithm  $\mathcal{A}$  scales with the size. In particular, if the maximum degree of any node in  $G$  is bounded, say by  $d$ , then each partition has at most  $d^K$  nodes. Then the overall cost is  $O(Q(d^K)p)$ , where  $Q(\ell)$  is the computation cost of  $\mathcal{A}$  for any graph with  $\ell$  vertices.

## IV. MAIN RESULTS

### A. GRAPHS WITH POLYNOMIAL GROWTH

We state sharp results for graphs with polynomial growth. We state results for MAP inference and for modularity optimization under the same theorem statement to avoid repetition. The proofs, however, will have some differences.

*Theorem 1:* Let the graph  $G = (V, E)$  have polynomial growth with degree  $\rho \geq 1$  and constant  $C \geq 1$ . Then, for a given  $\delta \in (0, 1)$ , select parameters

$$K = K(\rho, C, \delta) = \frac{8\rho}{\varepsilon} \log\left(\frac{8\rho}{\varepsilon}\right) + \frac{4}{\varepsilon} \log C + \frac{4}{\varepsilon} \log \frac{1}{\varepsilon} + 2,$$

$$\varepsilon = \varepsilon(\rho, C, \delta) = \begin{cases} \frac{\delta}{2C2^\rho}, & \text{for MAP} \\ \frac{\delta}{4(2C-1)}, & \text{for modularity optimization.} \end{cases} \quad (6)$$

Then, the following holds for the meta-algorithm described in Section III.

- (a) If  $\mathcal{A}$  solves the problem (MAP or modularity optimization) exactly, then the solution produced by the algorithm  $\hat{\mathbf{x}}$  and  $\hat{\chi}$  for MAP and modularity optimization respectively are such that

$$(1 - \delta)\mathcal{H}(\mathbf{x}^*) \leq \mathbb{E}[\mathcal{H}(\hat{\mathbf{x}})] \leq \mathcal{H}(\mathbf{x}^*)$$

$$(1 - \delta)\mathcal{M}(\chi^*) \leq \mathbb{E}[\mathcal{M}(\hat{\chi})] \leq \mathcal{M}(\chi^*). \quad (7)$$

- (b) If  $\mathcal{A}$  is an  $\alpha(n) \geq 1$  approximation algorithm for graphs with  $n$  nodes, then

$$\left(\frac{1}{\alpha(\tilde{K})} - \delta\right) \mathcal{H}(\mathbf{x}^*) \leq \mathbb{E}[\mathcal{H}(\hat{\mathbf{x}})] \leq \mathcal{H}(\mathbf{x}^*),$$

$$\frac{(1 - \delta)}{\alpha(\tilde{K})} \mathcal{M}(\chi^*) \leq \mathbb{E}[\mathcal{M}(\hat{\chi})] \leq \mathcal{M}(\chi^*), \quad (8)$$

where  $\tilde{K} = CK^\rho$ .

### B. GENERAL GRAPH

The theorem in the previous section was for graphs with polynomial growth. We now state results for a general graph.

Our result tells us how to evaluate the “error bound” on solutions produced by the algorithm for any instantiation of randomness. The result is stated below for both MAP and modularity optimization. The “error function” depends on the problem.

*Theorem 2: Given an arbitrary graph  $G = (V, E)$  and our algorithm operating on it with parameters  $K \geq 1$ ,  $\varepsilon \in (0, 1)$  using a known procedure  $\mathcal{A}$ , the following holds:*

- (a) *If  $\mathcal{A}$  solves the problem (MAP or modularity optimization) exactly, then the solution produced by the algorithm  $\widehat{\mathbf{x}}$  and  $\widehat{\chi}$  for MAP and modularity optimization respectively are such that (with  $\mathcal{B} = E \setminus \cup_{k=1}^p E_k$ ),*

$$\begin{aligned} \mathcal{H}(\widehat{\mathbf{x}}) &\geq \mathcal{H}(\mathbf{x}^*) - \sum_{(i,j) \in \mathcal{B}} (\psi_{ij}^U - \psi_{ij}^L), \\ \mathcal{M}(\widehat{\chi}) &\geq \mathcal{M}(\chi^*) - \frac{|\mathcal{B}|}{2m}. \end{aligned} \quad (9)$$

- (b) *If  $\mathcal{A}$  is instead an  $\alpha(n)$ -approximation for graphs of size  $n$ , then*

$$\begin{aligned} \mathcal{H}(\widehat{\mathbf{x}}) &\geq \frac{1}{\alpha(\tilde{K})} \left( \mathcal{H}(\mathbf{x}^*) - \sum_{(i,j) \in \mathcal{B}} (\psi_{ij}^U - \psi_{ij}^L) \right) \\ \mathcal{M}(\widehat{\chi}) &\geq \frac{1}{\alpha(\tilde{K})} \left( \mathcal{M}(\chi^*) - \frac{|\mathcal{B}|}{2m} \right), \end{aligned} \quad (10)$$

where  $\tilde{K}$  is the maximum number of nodes that are within  $K$  hops of any single node in  $V$ .

In the expression above,  $\psi_{ij}^U \triangleq \max_{\sigma, \sigma' \in \Sigma} \ln \psi_{ij}(\sigma, \sigma')$ , and  $\psi_{ij}^L \triangleq \min_{\sigma, \sigma' \in \Sigma} \ln \psi_{ij}(\sigma, \sigma')$ .

### C. DISCUSSION OF RESULTS

Here we dissect the implications of the above stated theorems. To start with, Theorem 1(a) suggests that when graphs have polynomial growth, there exists a Randomized Polynomial Time Approximation Scheme (PTAS) for MAP computation and modularity optimization that has computation time scaling linearly with  $n$ .

Theorem 1(b) suggests that if we use an approximation algorithm instead of the exact procedure for each partition, the resulting solution almost retains its approximation guarantees: if  $\alpha(n)$  is a constant, then the resulting approximation guarantee is essentially the same constant; if  $\alpha(n)$  increases with  $n$  (e.g.,  $\log n$ ), then the resulting algorithm provides a constant factor approximation! In either case, even if the approximation algorithm has superlinear computation time in the number of nodes (e.g., semi-definite programming), then our algorithm provides a way to achieve similar performance but in linear time for polynomially growing graphs.

The algorithm, for general graphs, produces a solution for which we have approximation guarantees. Specifically, the error scales with the fraction of edges across partitions that are induced by our partitioning procedure. This error depends on parameters  $K$ ,  $\varepsilon$  utilized by our partitioning procedure. For a graph with polynomial growth, we provide recommendations on what the values should be for these parameters. However, for general graphs, one may try various

values of  $K \in \{1, \dots, n\}$  and  $\varepsilon \in (0, 1)$  and then choose the best solution. Indeed, a reasonable way to implement such procedure would be to take values of  $K$  that are  $2^k$  for  $k \in \{0, \dots, \log n\}$  and  $\varepsilon$  chosen at regular interval with granularity that an implementor is comfortable with (the smaller the granularity, the better).

The dependence on  $\rho$  and  $\delta$  in Theorem 1 is inspired by the worst-case scenario. While they provide theoretically useful bounds, practically even for moderately small  $\delta$ , it may be an exorbitant amount of computation required if we followed the guidelines of Theorem 1 to implement brute-force/dynamic programming procedure. However, in practice, a smaller radius than that suggested by Theorem 1 can lead to better performance, as observed in experiments.

### V. EXPERIMENTAL SETUP

In Sections V and VI, we present the experimental evaluations of our algorithm regarding two questions of interest: computation of Maximum A Posteriori (MAP) inference in a pairwise Markov Random Field (MRF), and modularity optimization for community detection. For our experiments, we applied PM both on real-world networks and on synthetic networks to verify its efficiency compared to the original centralized algorithm. For the MAP inference, we employed the sequential tree-reweighted max-product message passing (TRW-S) as the centralized algorithm. For the modularity optimization, Girvan-Newman (GN), Clauset-Newman-Moore (CNM), and Louvain-Method (LM) have been used as the centralized algorithm.

In the experiments, we fixed the radius (for every iteration) by a specific number to simplify and to understand the performance of our algorithm according to the radius. Furthermore, we investigated the value of partition radius as an appropriate value for our algorithm. All experiments were carried out using a single core<sup>3</sup> of an Intel i7 processor with 256GB of RAM, and PM was implemented in C++.

### A. DATASETS

#### 1) REAL NETWORKS

To cover various aspects of the real-world networks, we conducted experiments on several types of real-world networks, such as social networks (Facebook, YouTube, Live Journal, Twitter, Friendster), citation or collaboration networks (ArXiv, DBLP), and web or communication networks (Email-Enron, Web uk-2005, Web Webbase-2001). Most of the real-world network datasets were downloaded from SNAP,<sup>4</sup> a web site that offers the information and statistics of the networks. The network statistics are summarized in Table 1. Note that the list below the dashed line is large-scale networks with more than 10 million nodes. For a directed real-world network, we used the corresponding undirected network in our experiments.

<sup>3</sup>When you apply this method in parallel, you can easily allocate different graph partitions to different cores/computers. This is because PM algorithm divides the whole graph into small disjoint subgraphs which can be computed almost independently.

<sup>4</sup>SNAP(<http://snap.stanford.edu>)

TABLE 1. Statistics for real-world networks.

Networks	Nodes	Edges	Average Degree	Average Distance	Diameter	90th Effective Diameter
Facebook	4,039	88,234	43.69	3.7	8	4.7
ArXiv	9,377	24,107	5.14	6.8	19	8.7
Email-Enron	36,692	183,831	10.02	3.2	13	4.8
DBLP	317,080	1,049,866	6.62	5.6	22	8.1
YouTube	1,134,890	2,987,624	5.27	4.2	15	4.9
Live Journal	3,997,962	34,681,189	17.35	4.9	17	6.5
<hr/>						
Web uk-2005	39M	798M	39.69	7.5	25	9.3
Twitter	41M	1.2B	57.74	3.6	18	4.1
Friendster	65M	1.8B	55.06	5.1	32	5.8
Web Webbase-2001	118M	1B	14.78	8.5	383	10.9

## 2) SYNTHETIC GRAPHS

We also made use of synthetically generated graphs such as grid graphs, random  $k$ -regular graphs, Watts-Strogatz networks, and Barabási-Albert networks.

### a: GRID GRAPHS

We tested our algorithm with two types of grid graphs. One is a simple grid, whose nodes are connected if they are directly adjacent to each other in either the horizontal or the vertical direction; the other one is the grid graph with diagonal edges.

### b: RANDOM $k$ -REGULAR NETWORKS

A random  $k$ -regular network on  $n$  nodes,  $G_{n,k}$ , is a random graph chosen uniformly at random from all the graphs whose nodes have degree  $k$ .

### c: WATTS-STROGATZ NETWORKS

To construct Watts-Strogatz, we start from a simple grid of  $n$  nodes. First, each node is connected to the other nodes within a radius of  $r$ , which represents the length of the shortest path between the locally connected nodes. Next, we add  $l$  long distance edges, one end node of which remains the same and the other chosen uniformly at random from among all nodes. This process is repeated for each node in the network.

### d: BARABÁSI-ALBERT NETWORKS

We begin with the Erdős-Rényi model of  $m_0$  nodes with edge probability  $p$ , which will be defined below. In other words, given  $m_0$  nodes, each node-pair is connected with probability  $p$  independent of every other node-pair. Then, the network is developed following an iterative process until the total number of nodes becomes  $n$ . At each discrete time step, we add a new node, and each new node is connected to  $m$  existing nodes with a probability that is proportional to the degree of each node. We require  $m$  to be  $m_0$  times  $p$ .

## B. CENTRALIZED ALGORITHMS

To facilitate understanding of the effectiveness of our algorithm, we selected TRW-S for MAP inference and three popular community detection algorithms for modularity optimization, which are taken as a centralized algorithm for PM. The algorithms used for our experiments are briefly summarized as follows:

### 1) MAP INFERENCE

#### a: SEQUENTIAL TREE-REWEIGHTED MAX PRODUCT MESSAGE PASSING (TRW-S)

The TRW-S algorithm is used for minimizing energy functions of discrete variables with unary and pairwise terms. It is the modified version of TRW that is not guaranteed to increase a lower bound on the energy and does not always converge. By adding a weak tree agreement (WTA) condition, TRW-S guarantees to find at least a local maximum of the bound, and it has a subsequence converging to a vector satisfying WTA [10].

#### b: ENERGY FUNCTION

For the MAP inference, we considered an energy function with binary labels (i.e.  $\Sigma = \{0, 1\}$ ) on a graph  $G = (V, E)$ :

$$\mathbb{E}(\mathbf{x}) = \sum_{i \in V} \theta_i x_i + \sum_{(i,j) \in E} \theta_{ij} x_i x_j, \quad \text{for } \mathbf{x} \in \{0, 1\}. \quad (11)$$

We consider the following scenario for choosing parameters, where  $\mathcal{U}[a, b]$  is the uniform distribution over the interval  $[a, b]$ :

1. For each  $i \in V$ , choose  $\theta_i$  independently as per the distribution  $\mathcal{U}[-1, 1]$ .
2. For each  $(i, j) \in E$ , choose  $\theta_{ij}$  independently from  $\mathcal{U}[-\alpha, \alpha]$ . Here, the *interaction* parameter  $\alpha$  is chosen from  $\{0.25, 1, 4, 16\}$ .

## C. MODULARITY OPTIMIZATION

### 1) GIRVAN AND NEWMAN (GN)

The GN algorithm is the first known modularity-based method for community detection. Starting from the original network, the edges are iteratively removed to uncover the underlying community structure of the network. This process is based on the concept of the edge betweenness, which represents the number of shortest paths between pairs of nodes that pass through the edge, and it is repeated until no edges remain. The algorithm is somewhat slow and has a computational complexity  $O(N^3)$  on a sparse network [11].

### 2) CLAUSET-NEWMAN-MOORE (CNM)

The CNM algorithm, which utilizes efficient data structures such as a max-heap, is a kind of fast version of Girvan-Newman algorithm. Every node begins as a single



community of the network. At each time step, the two communities that yield the largest increase of modularity among all the pairs of communities are combined together. The algorithm allows analysis of the community structure of large graphs, up to  $10^6$  nodes, with a computational complexity  $O(n \log^2 n)$  on a sparse network [12].

### 3) LOUVAIN METHOD (LM)

The LM algorithm is known as one of the best for detecting community structure. As a state-of-the-art algorithm, it finds good divisions in terms of modularity even on large networks, and it reveals a hierarchical community structure that is based on the two-step sequential processes (local maximization of the modularity and aggregation of communities). The algorithm is fast enough to be limited in its applicable network size due to restricted storage capacity rather than restricted computation time, with a computational complexity that is essentially linear to network size [13].

## VI. EXPERIMENTAL RESULTS

As a measure of performance, we compared (i) energy and running time for MAP inference, and (ii) modularity and running time for modularity optimization, calculated by our algorithm, against those obtained by using the original centralized algorithms. In addition, we investigated what value of partition radius can be determined as appropriate for our algorithm, which will be described in Subsection VI-B. We lay out the experimental results in several changes of the network factors to determine the conditions under which our algorithm performs well.

For MAP inference, the simulation was carried out over the 30 different samples of the same problem. Then, the average energy was computed over those samples for 100 trials for each case. While the average value obtained from our setup is a negative quantity, we report it as a non-negative value regardless of its sign for the unity of expression. For modularity optimization, the average modularity is simply calculated for the 100 trials for each case.

This section is organized as follows: Section VI-A presents the overall result of the experiment. Section VI-B describes how to obtain the proper partition radius. Sections VI-C and VI-D present the analyses of the experimental results on real-world networks and synthetic graphs, respectively. Section VI-E describes the results and their explanation in regard to the two problems.

*Plot:* To be conducive to performance comparison, we used a ratio of energy/modularity and running time, and these quantities are plotted as functions of the partition radius. In other words, the partition radius is plotted on the x-axis, the ratio of the results of PM algorithm to that of the centralized algorithm on the y-axis.

### A. OVERALL RESULTS

Overall, PM provides a decent trade-off between high accuracy and low complexity, with particularly good efficiency when a proper partition radius is chosen. That is, our algorithm produces a good approximation in a relatively short

time for most networks. In particular, PM substantially reduces the running time for the centralized algorithms with high computational complexity, while energy/modularity remains at similar levels – even better in some cases – to that of the centralized algorithms.

To sum up, PM performs better under the following conditions: (i) when applied on well-distributed regular networks; (ii) when the centralized algorithm has a high complexity; and (iii) generally when the network has a large size.

In some cases, PM shows different tendencies towards the two problems we dealt with, even under the same conditions, which is due to the two problems being different: MAP inference is based on assignment, and modularity optimization is based on graph clustering. Furthermore, we empirically prove that modularity optimization are indeed decomposable problems as long as the network has geometric structures.

### 1) ENERGY AND MODULARITY

Energy/Modularity typically tends to increase along with the partition radius. Our partitioning scheme involves the removal of the edges that are not included in any partition. As we have previously proved, the experimental results demonstrate that the error actually scales with the fraction of edges across partitions. That is, as partition radius increases, the number of edges crossing partition decreases. Accordingly, the error is reduced, and we can take into account the connectivity between more nodes. This leads us to find better assignment/division, which means an increase in energy/modularity.

### 2) RUNNING TIME

As shown in Section III, the overall computation cost increases as an exponential function of radius. However, some experimental results show that this is not the only case. One exceptional case can be observed when large numbers of partitions are generated due to a small partition radius. We assume that this situation can cause a great deal of wall-clock time. The other case is when applying to the network with hubs (e.g., Barabási-Albert network), the node with a very high degree. In such cases, each partition has a large number of nodes even when the partition radius is small. In exceptional cases, the performance of our algorithm is relatively poor for modularity optimization, while PM shows excellent performance for MAP inference. In particular, for MAP inference, PM produces similar results to the centralized algorithms in less than half the time, with a small partition radius. In our experiments, the time required for the partitioning procedure is just within 0.1–0.2 seconds, even on the network of  $10^6$  nodes, and it is so tiny as to be almost insignificant in total running time. In addition, it should be noted that, even if the efficiency of PM decreases, the actual gain of running time generally increases due to the significant growth in the running time of the centralized algorithms.

### B. PARTITION RADIUS

The primary question of interest: “*what value of partition radius should we choose?*” Theoretical results presented

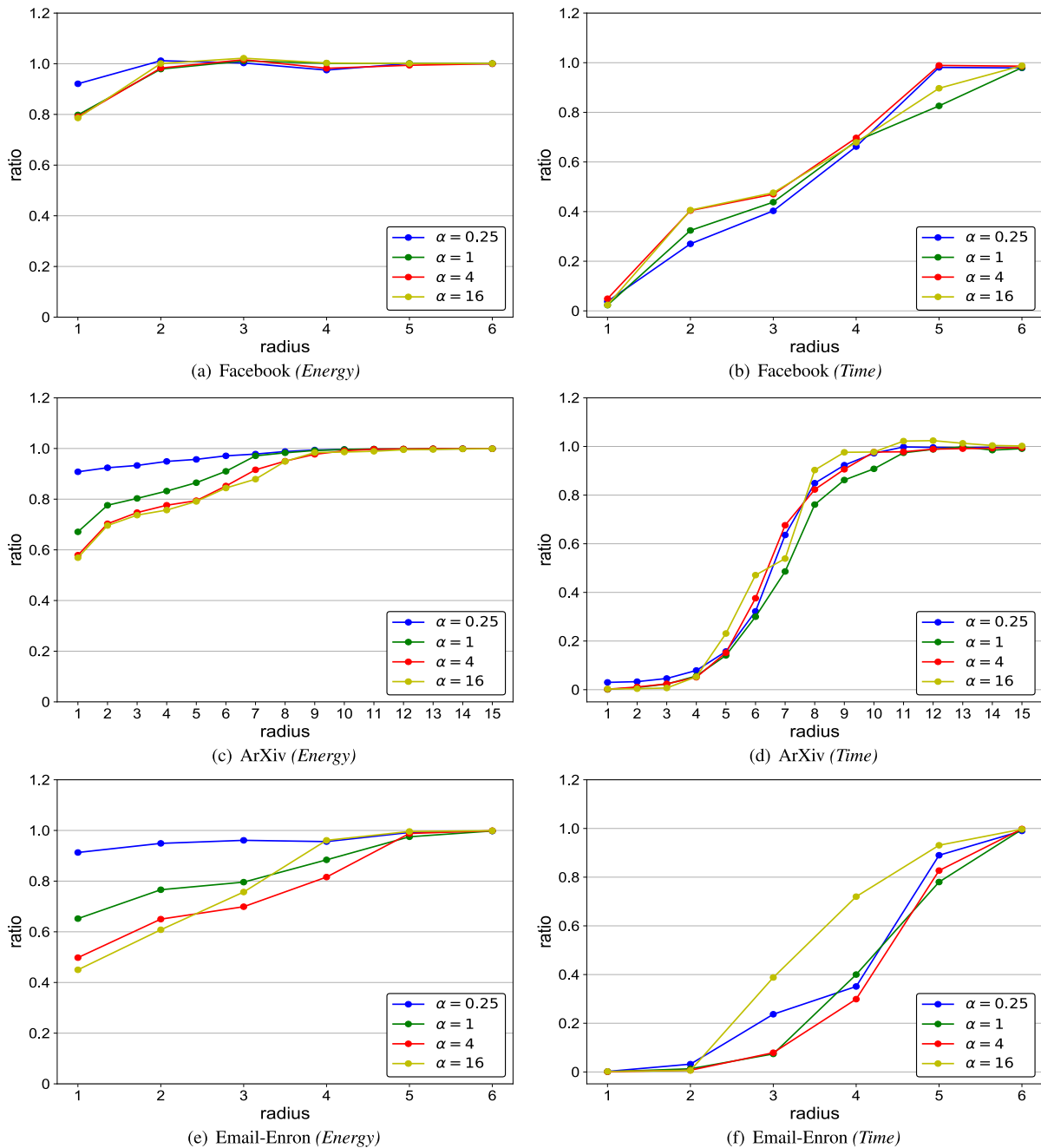


FIGURE 2. Small-scale real-world networks for MAP inference.

earlier in this work provide guidelines. It is definitely worth following. In addition, practically, we find a simple heuristic rule works well for “regular” enough graphs. To that end, define the average distance of a graph as the average shortest path length between all pairs of nodes. To estimate it, one can simply sample 1,000 random node pairs and calculate the average over them. Then the following is a heuristic we suggest to choose partition radius:

*choose  $\lceil \text{Avg}.D \rceil$  or  $\lceil \text{Avg}.D \rceil - 1$  as the partition radius.*

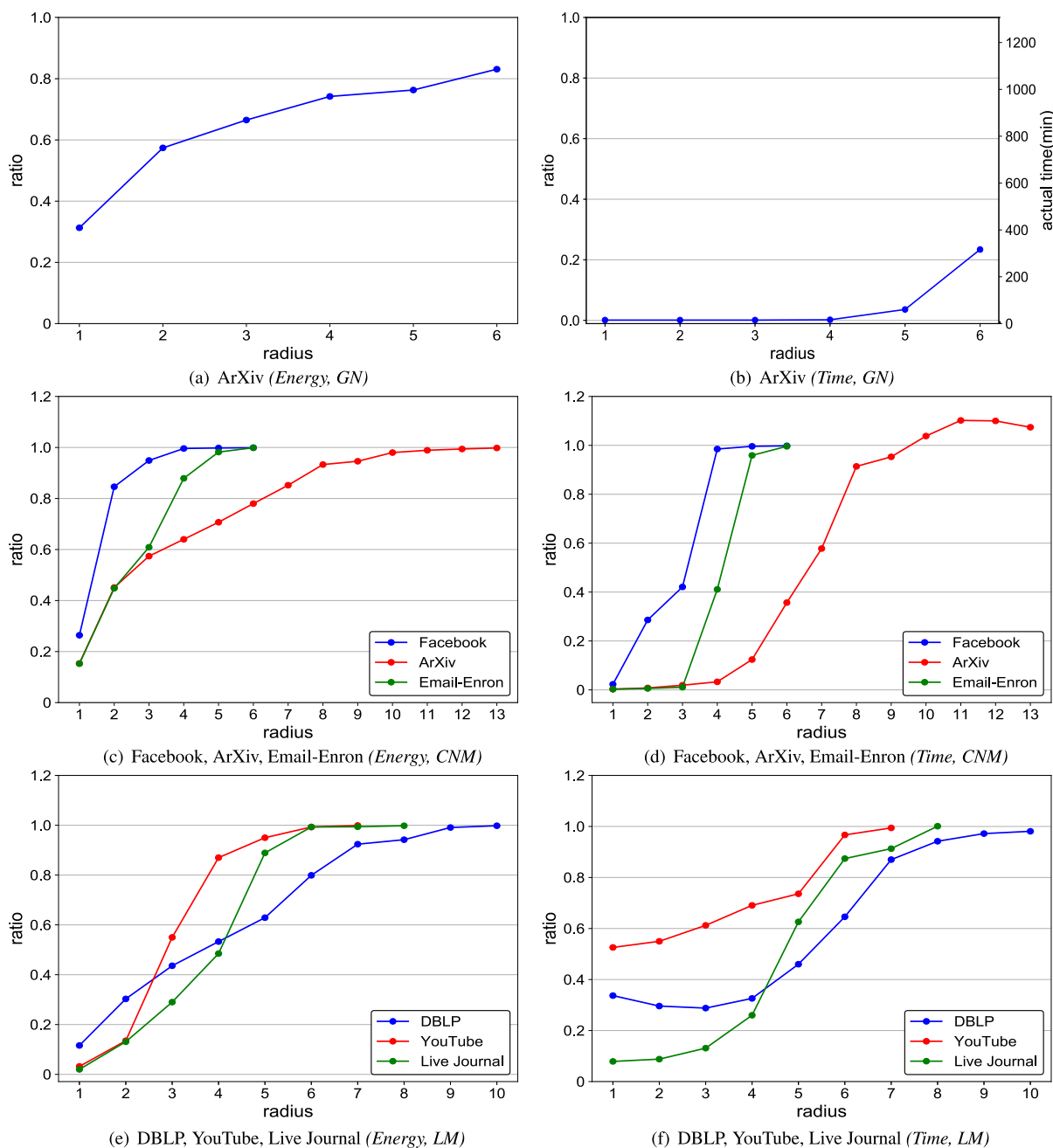
The intuition behind this rule is as follows. If the graph is “regular” enough, then the pair-wise distance distribution

between nodes is “uni-modal”, and in which case the average distance is precisely that covers good fraction of interactions. It may make sense to choose a little smaller radius (by 1) if the graph has few very high degree nodes.

### C. ON REAL-WORLD NETWORKS

#### 1) ON SMALL-SCALE NETWORKS

Figures 2 and 3 show the experimental results from applying our algorithm on the small-scale real-world networks. In Figures 2(a) and 2(b), on Facebook, PM finds solutions with higher energy than those obtained by using the original



**FIGURE 3. Small-scale real-world networks for modularity optimization. In Figure 3(b), the actual running time (unit: minute) can be read out from the right y-axis. Note that it takes 22 hours for GN to analyze the ArXiv network.**

TRW-S within half the time in regards to MAP inference. For modularity optimization, on the same network, PM also achieves similar modularity to the original CNM in a relatively short time, as shown in Figures 3(c) and 3(d). These particularly good results are due to the very high average degree (evenly distributed) of Facebook nodes (about 43). Note that in the case of  $R(\text{Avg.D}) - 1$ , which is smaller than the value of general cases, it could be a better choice for the partition radius. ArXiv has an intractable size for GN. Although it requires a long time to analyze large partitions, solving the problem inside small partitions brings decent results.

This is possible because the network is well distributed. Figures 3(a) and 3(b) shows that PM gives around 78% modularity of GN in about 50 minutes. Considering that it takes about 22 hours for GN to apply on the entire network, this is a decent result. On the whole, PM shows good results on small-scale real-world networks.

## 2) ON LARGE-SCALE NETWORKS

When applying algorithms to large-scale graphs, PM is the most necessary moment. To verify our algorithm on large-scale graphs, we carried out the experiments on four

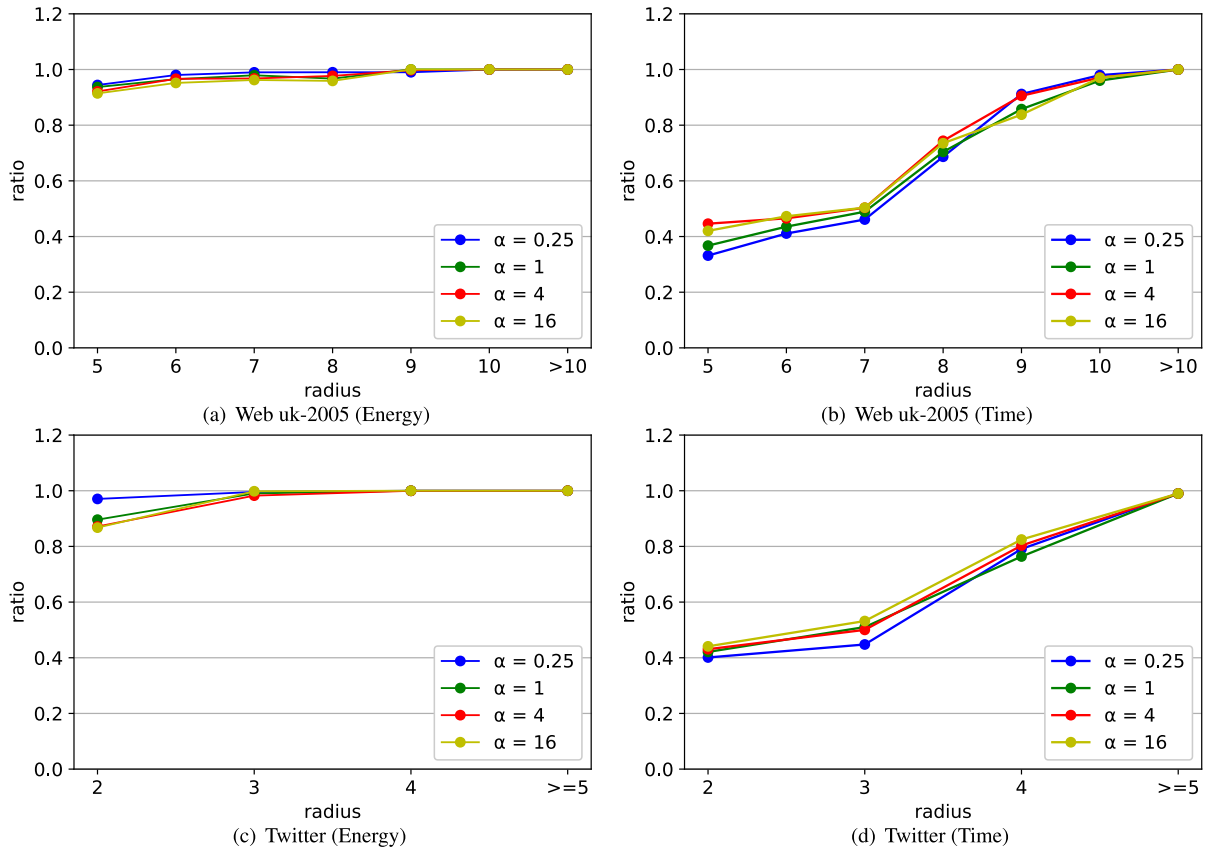


FIGURE 4. Large-scale real-world networks for MAP inference.

real-world networks with more than 10 million nodes. We used the TRW-S and LM as the centralized algorithm for MAP inference and modularity optimization, respectively. Note that we present the results centering on the partition radius we suggested in Section VI-B. As shown in Figure 4, PM shows outstanding results on MAP inference. When we choose the partition radius as  $\lceil Avg.D \rceil - 1$ , PM produces the same level of results as the centralized algorithm (TRW-S) in less than half the time. We analyze that these results are attributable to the high average degree of the network (the average degree of Web uk-2005 and Twitter is 40 and 58, respectively). Our algorithm involves the process of erasing the edges, but in this case, each node still has many edges that can refer to other nodes. We can see a similar trend on the Facebook network in Figure 2(a). Moreover, the original TRW-S algorithm takes more than 24 hours to compute the results, but PM makes the application of the algorithm more practical. For modularity optimization, our algorithm inevitably causes the loss in that modularity is the metric to be directly affected by removing the edges. Nevertheless, PM shows a good trade-off between modularity and running time, as shown in Figure 5.

D. ON SYNTHETIC GRAPHS

1) ON GRID GRAPHS

PM shows a tremendous performance on the types of grid graphs stated in Section V-A. For MAP inference,

Figure 6 demonstrates that our algorithm produces a similar value of the energy to the original TRW-S in about 1–2 percent of the time on the simple grid of  $10^6$  nodes. With diagonal edges, it takes about 6–7 percent of the time under the same conditions, as shown in Figure 7. The substantial decrease in running time underlines the effectiveness of our algorithm and strongly supports the conclusion that PM performs well on regular networks. In addition, the increase in the number of neighbor nodes makes networks more complex, which leads to large errors and, thus, a decrease in the efficiency of PM. As shown in Figures 8 and 9, PM also yields good results for modularity optimization. Moreover, PM achieves a better performance on larger graphs.

2) RANDOM K-REGULAR NETWORKS

For MAP inference, PM shows almost the same efficiency for a fixed degree, irrespective of the network size. In Figure 10, the results show that PM gives around 80% of the energy of TRW-S in about 20% of the time in all cases. On the other hand, for modularity optimization, PM performs better as the network size increases, as shown in Figure 12. For the same size of networks, the increase in degree makes our algorithm less effective, as shown in Figures 11 and 13. These results indicate that increasing randomness and complexity of networks have detrimental effects on our algorithm. As is the case of grid graphs, PM produces better efficiency on well-distributed regular networks.



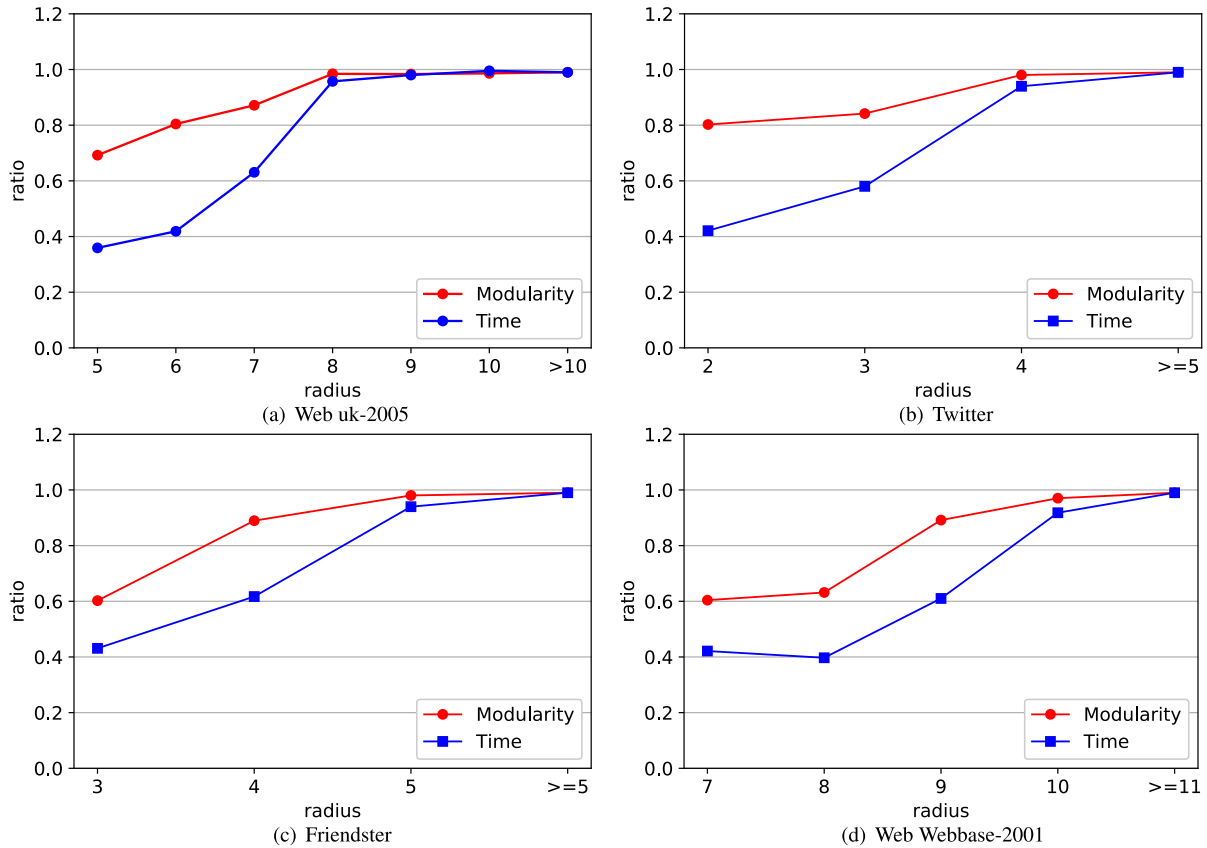


FIGURE 5. Large-scale real-world networks for modularity optimization.

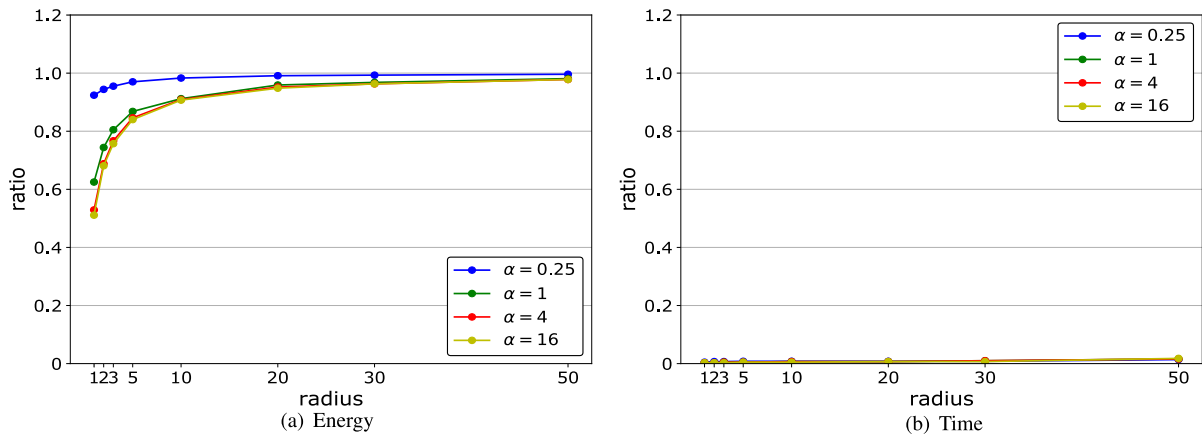


FIGURE 6. 1000-by-1000 grid graph for MAP inference.

### 3) ON WATTS-STROGATZ NETWORKS

Note that a Watts-Strogatz network has an underlying regular structure with a small amount of randomness. First, the randomness increases with the number of long-distance edges. Increasing randomness could have adverse effects on our algorithm. For both the problems, the experimental results are better for our algorithm with the small number of long-distance edges, as shown in Figures 14 and 16. In addition, long-distance edges reduce distributed processing

capability by making the networks denser, leading to deterioration in the efficiency of our algorithm.

On the other hand, two problems produce slightly different results regarding the change in the radius. For MAP inference, a large radius serves as an advantage for the small partition radius. This is attributed to the growth in the number of neighbor nodes, which is proportional to  $2rad^2$ . However, the performance deteriorates rapidly with the increase in the partition radius. In Figure 15, the results show that our

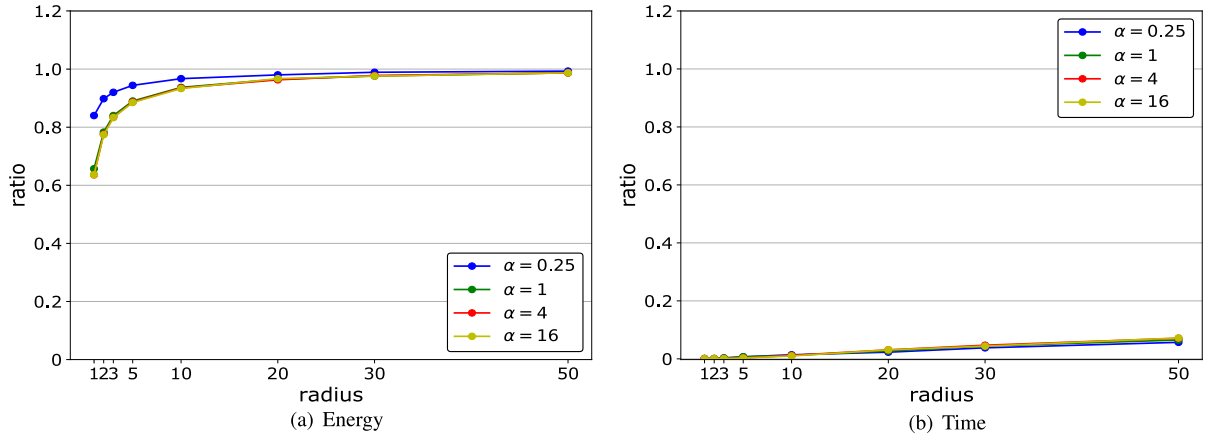


FIGURE 7. 1000-by-1000 grid graph with diagonal edges for MAP inference.

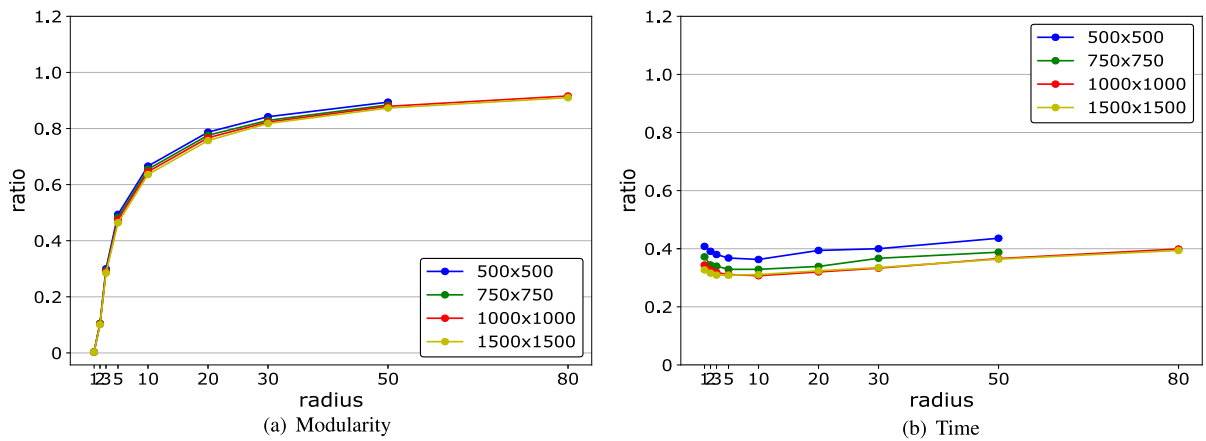


FIGURE 8. Grid graphs for modularity optimization (LM).

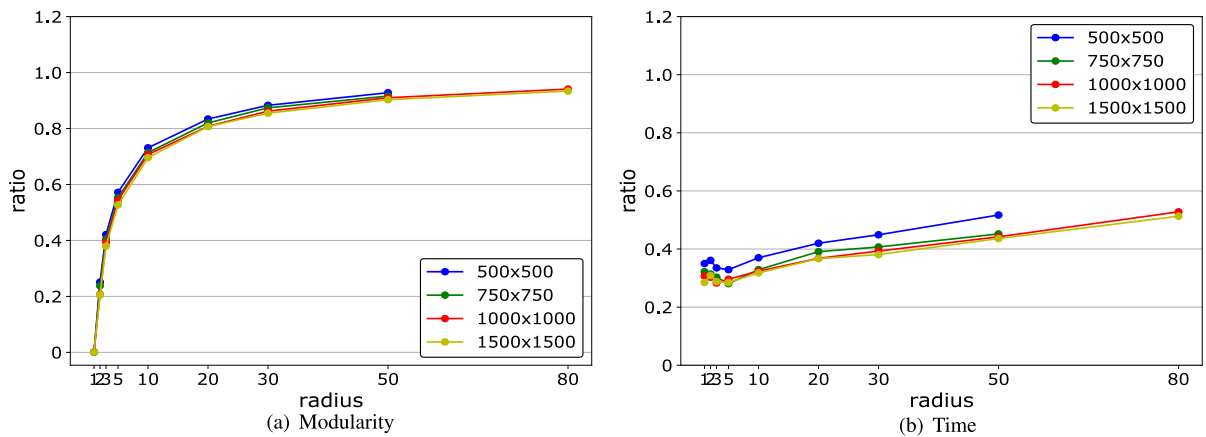


FIGURE 9. Grid graphs with diagonal edges for modularity optimization (LM).

algorithm is very efficient, producing nearly 81% energy compared to TRW-S within just 2% of the time for very small partition radii, while the performance is drastically degraded as the partition radius increases. For modularity optimization, the increase in radius vastly improves the efficiency of PM, as shown in Figure 17. The proximity-based connectivity is reinforced by a larger radius, offsetting the impact of

randomness and thereby promoting the efficiency of clustering. However, as is the case with long-distance edges, the increasing radius reduces the distributed effects.

#### 4) ON BARABÁSI-ALBERT NETWORKS

For both the problems, as the number of additional edges at each time step increases, the advantage of our

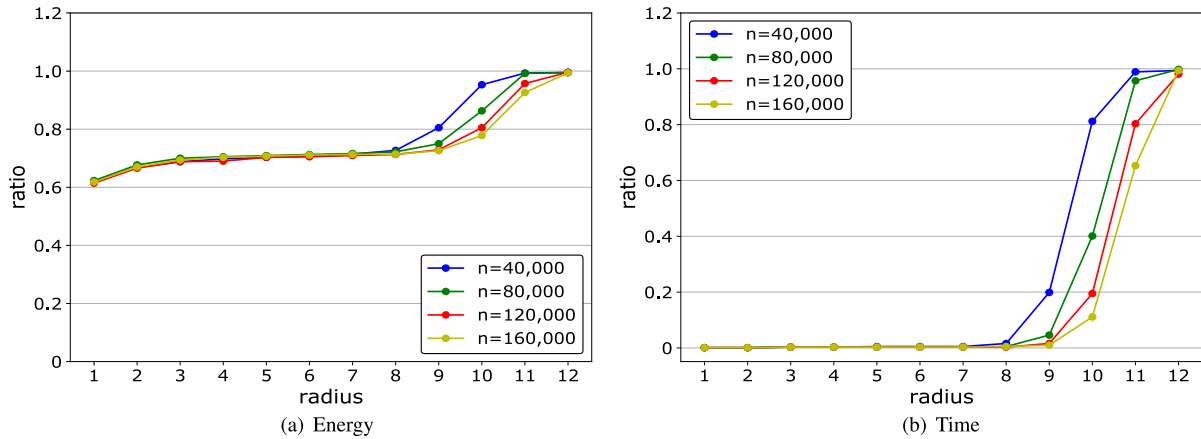


FIGURE 10. Random k-regular networks for MAP inference (For fixed  $k = 4, \alpha = 1$ ).

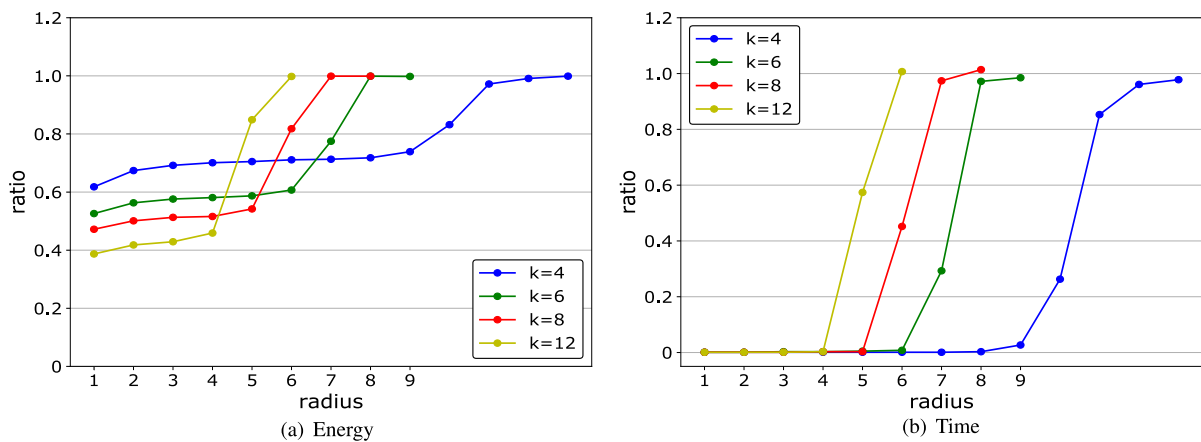


FIGURE 11. Random k-regular networks for MAP inference (For fixed  $n = 10^5, \alpha = 1$ ).

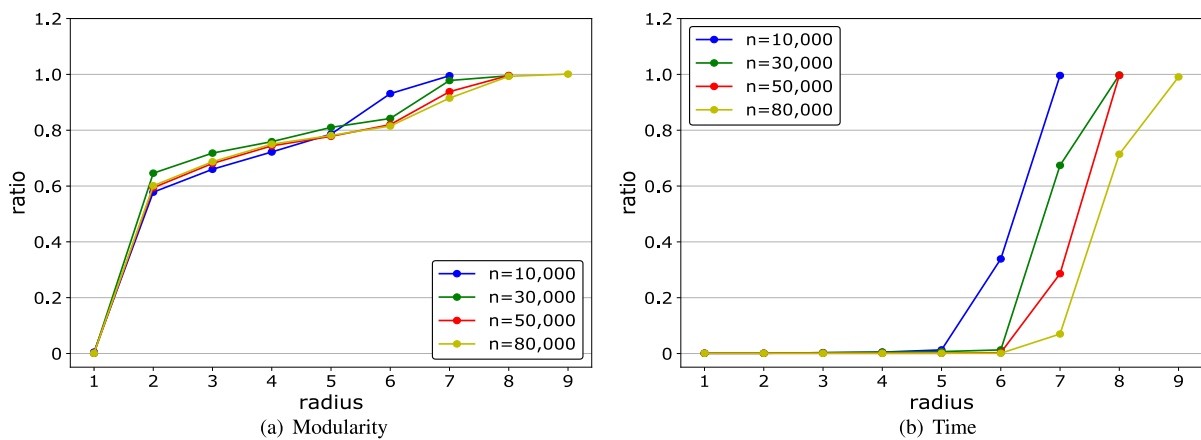


FIGURE 12. Random k-regular networks for modularity optimization (For fixed  $k = 6 / CNM$ ).

algorithm decreases. The results for this case are shown in Figures 18 and 19. Hubs could exist due to the generative processes of the Barabási-Albert network. These hubs significantly shorten the average distance between two nodes compared to the regular networks, such as grid graphs. For this reason, Barabási-Albert networks become

more complex and dense, and thus the efficiency of our algorithm, including the distributed effects, is reduced. For fixed additional edges, larger networks lead to better results. In Figure 20, the results show that increasing network size brings about more efficient distributed effects.

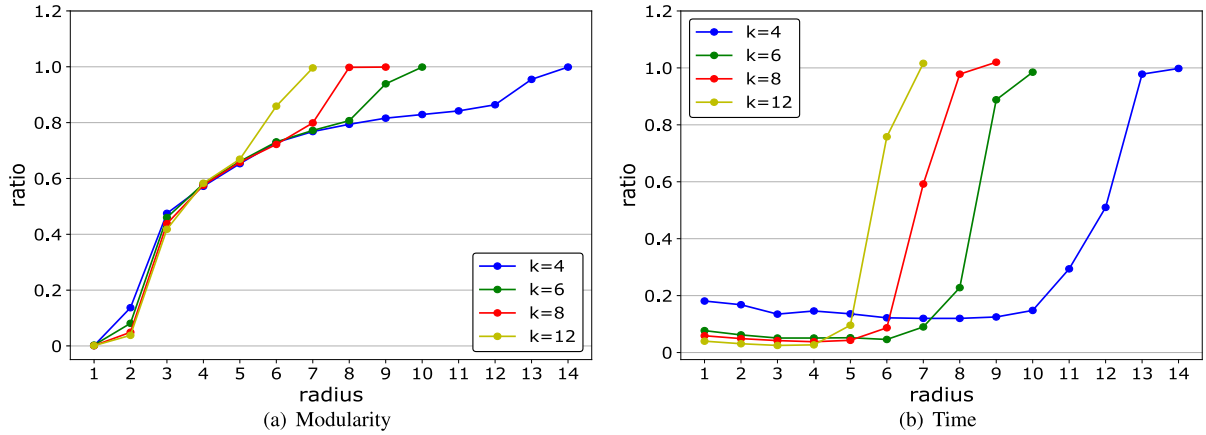


FIGURE 13. Random k-regular networks for modularity optimization (For fixed  $n = 10^6 / LM$ ).

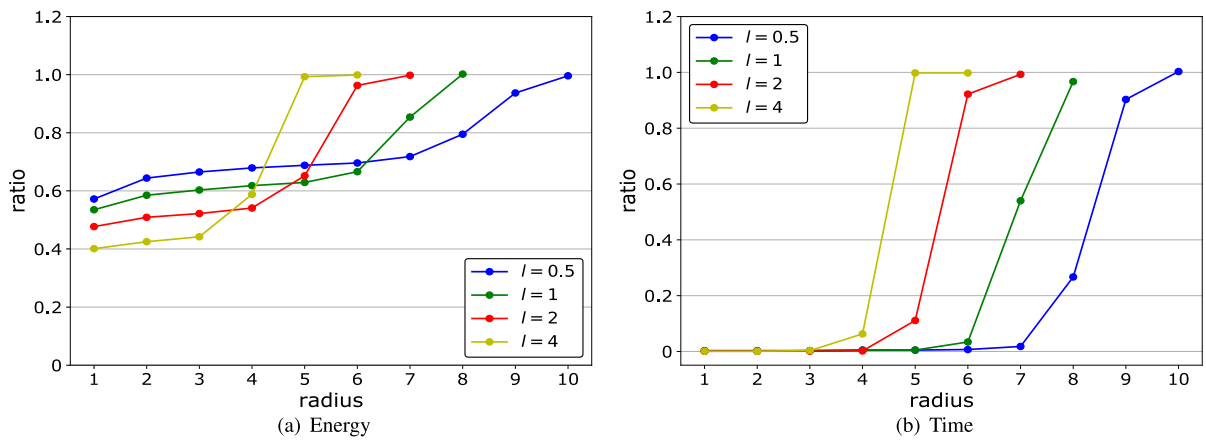


FIGURE 14. Watts-Strogatz networks for MAP inference (For fixed  $n = 300$ -by- $300$ ,  $r = 1$ ,  $\alpha = 1$ ).

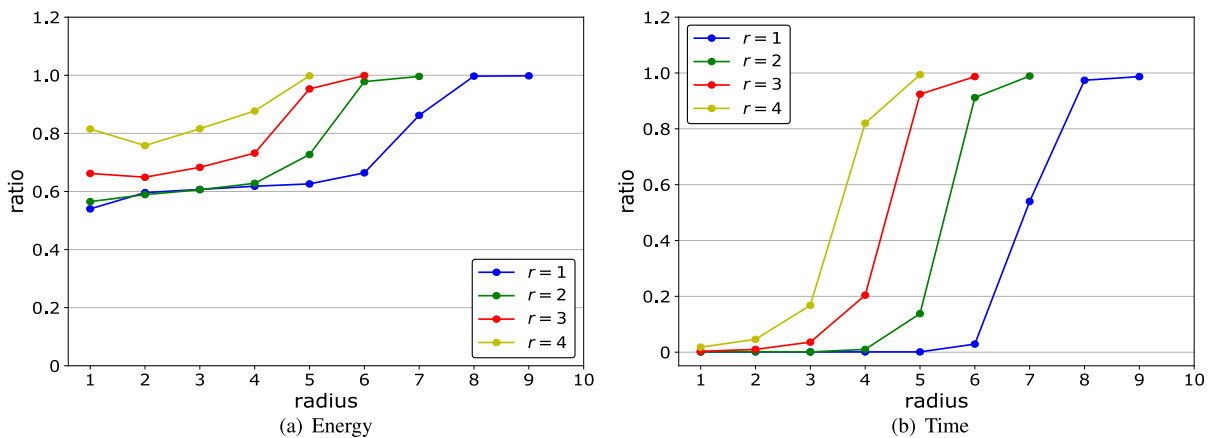


FIGURE 15. Watts-Strogatz networks for MAP inference (For fixed  $n = 250$ -by- $250$ ,  $l = 1$ ,  $\alpha = 1$ ).

**E. DISCUSSION FOR EACH PROBLEM**

1) MAP INFERENCE

$\alpha$ : INTERACTION PARAMETER  $\alpha$

We defined energy functions of discrete variables with unary and pairwise terms, where parameter  $\alpha$  of (11)

determines the strength of the relationship between pairs of nodes. Accordingly, the increasing value of  $\alpha$  assigns a larger weight to edges, which consequently incurs more errors caused by our partitioning scheme. Therefore, our algorithm generally yields better



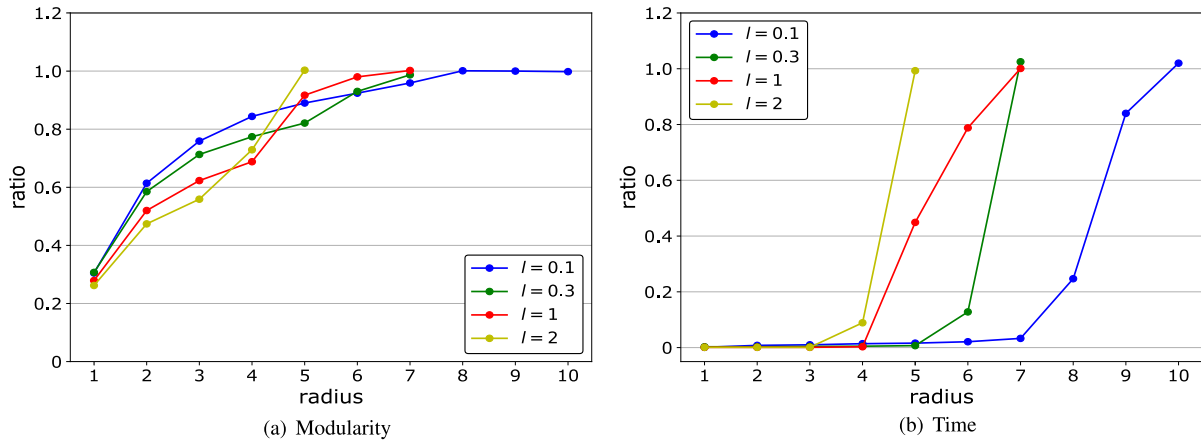


FIGURE 16. Watts-Strogatz networks for modularity optimization (For fixed  $n = 150$ -by- $150$ ,  $r = 2 / CNM$ ).

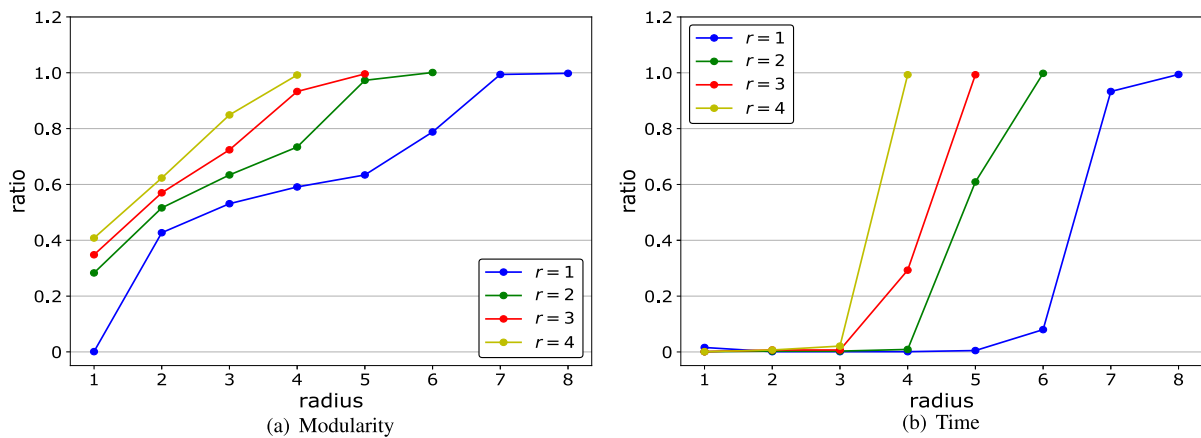


FIGURE 17. Watts-Strogatz networks for modularity optimization (For fixed  $n = 100$ -by- $100$ ,  $l = 1 / CNM$ ).

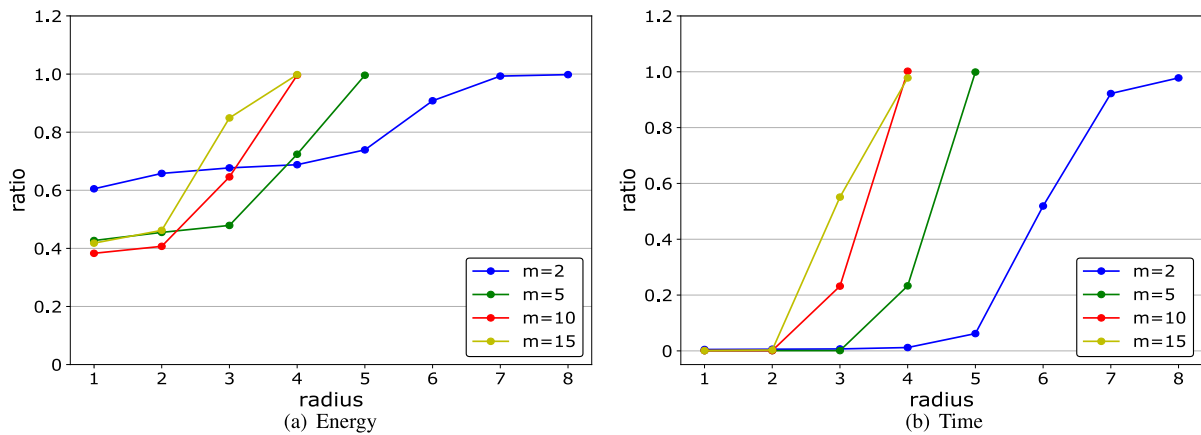


FIGURE 18. Barabási-Albert networks for MAP inference (For fixed  $n = 50,000$ ,  $\alpha = 1$ ).

approximation when  $\alpha$  has a relatively small value, as shown in Figure 21.

*b: AVERAGE DEGREE*

It is observed that MAP inference is more affected by the average degree than modularity optimization. When partitions are very small, there is a tendency to produce better

approximation as the average degree increases. In particular, when the average degree is very high, our algorithm obtains outstanding results even for tiny partitions, such as in Facebook, Web uk-2005, and Twitter. However, one cannot always obtain good results with a high average degree. As you can see throughout the experiments, the increase of average degree that leads to an increase in randomness can

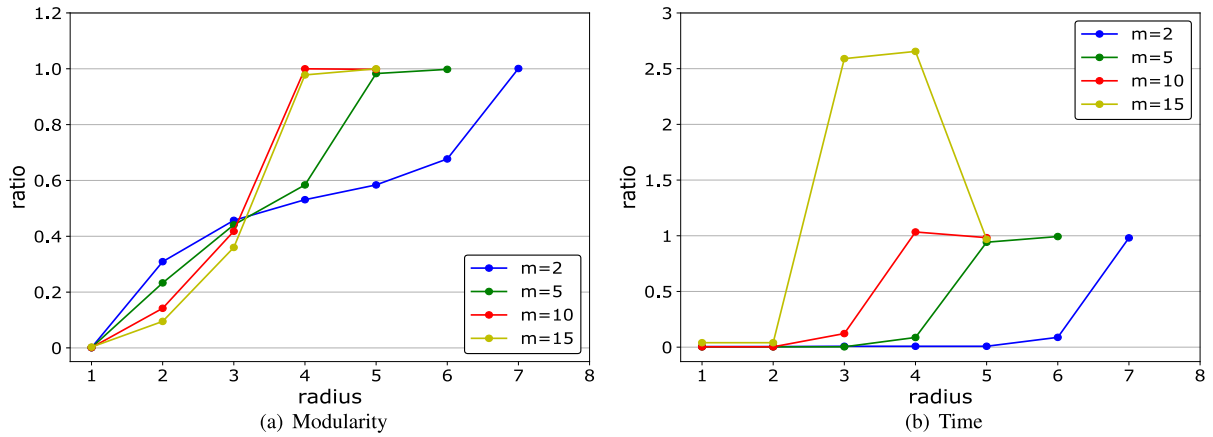


FIGURE 19. Barabási-Albert networks for modularity optimization (For fixed  $n = 10^6 / LM$ ).

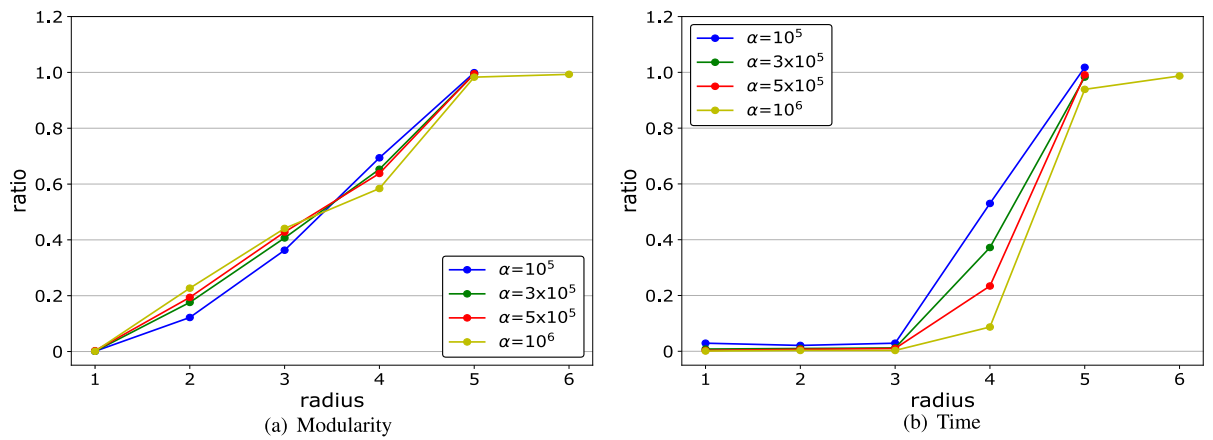


FIGURE 20. Barabási-Albert networks for modularity optimization (For fixed  $m = 5 / LM$ ).

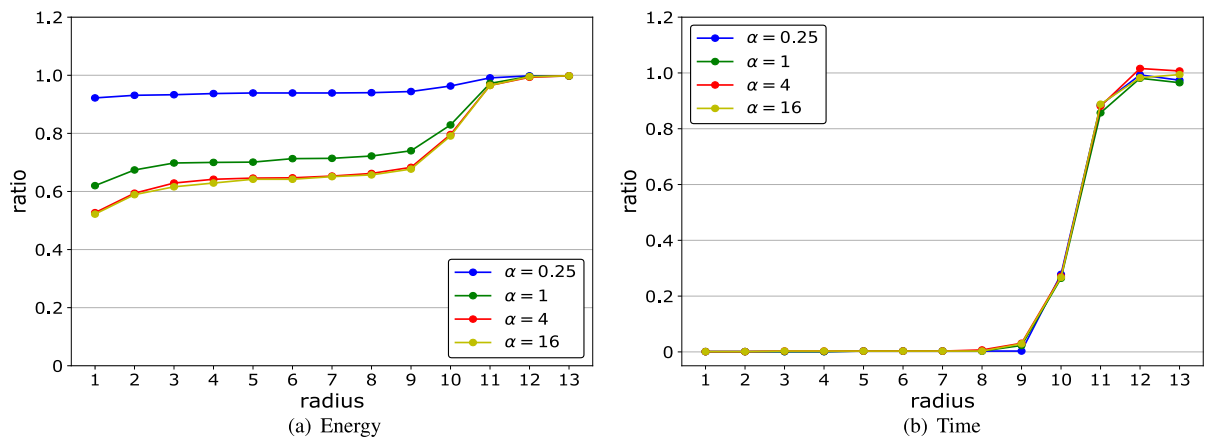


FIGURE 21. Results as the value of  $\alpha$  changes. Random  $k$ -regular networks for MAP (For fixed  $n = 10^5, k = 4$ ).

negatively influence our algorithm. Our algorithm accomplishes better efficiency generally when the increase of average degree improves the regularity. However, in like manner to  $\alpha$ , it requires relatively more time for better assignment as the partition radius increases when the average degree is high.

Taken together, the best circumstances for our algorithm in regards to MAP inference is when  $\alpha$  has a small value with a high average degree. However, our algorithm can only realize positive effects when the increase of the average degree improves the regularity. In other words, PM shows better results when applied on well-distributed regular networks

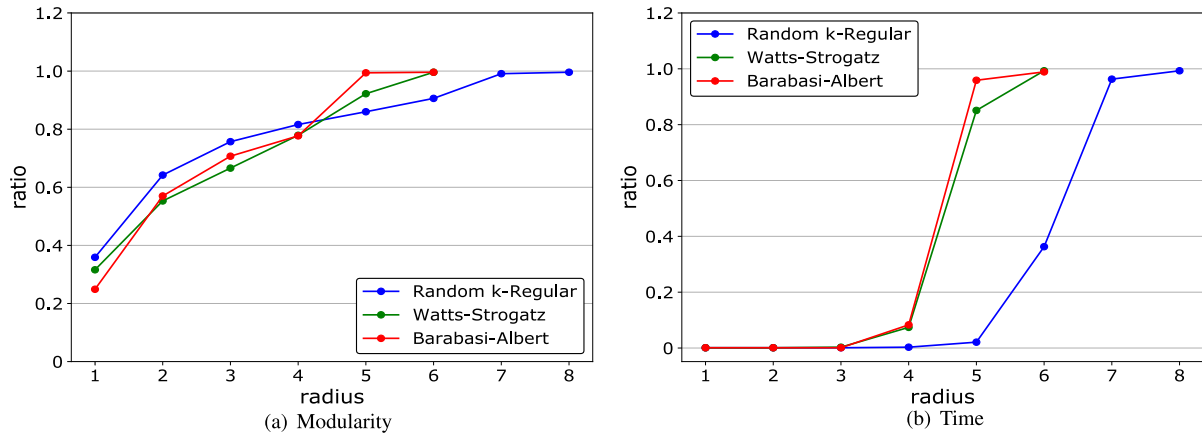


FIGURE 22. The experimental results for Girvan and Newman algorithm.

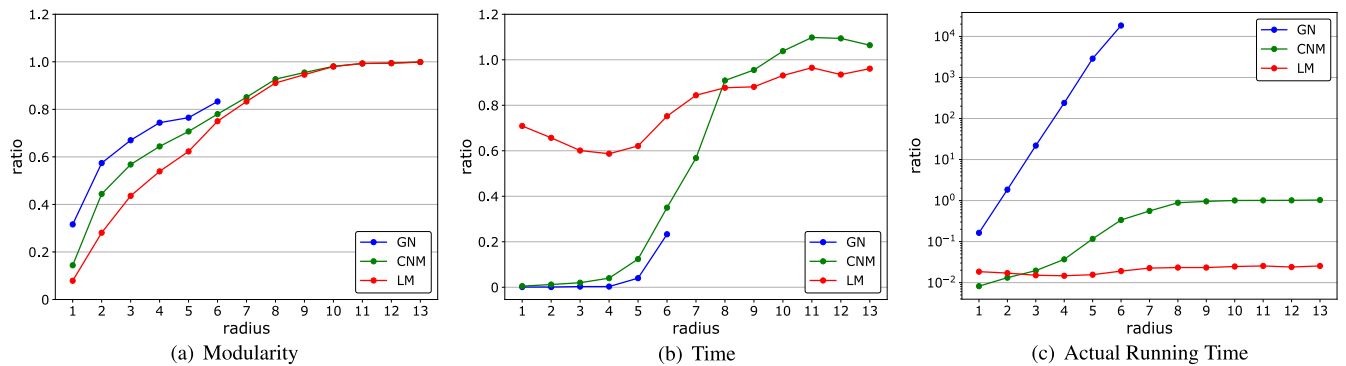


FIGURE 23. A Comparison on results of three algorithms for modularity optimization.

while taking less consideration of the relationship between nodes.

## 2) MODULARITY OPTIMIZATION

Due to its high computational complexity, it is difficult for GN to analyze networks under several conditions. Accordingly, we present the experimental results of GN separately, and PM shows a good performance in this case, as shown in Figure 22. Also, we observe that the three centralized algorithms we used to extract the community structure show somewhat different tendencies regarding the change in the partition radius. Focusing on these differences, we split the analysis of the experimental results into two parts for closer examination:

### a: MODULARITY

As shown in Figure 23(a), GN produces the largest modularity on the small partitions, followed by CNM and LM in order. This differs from the results we could get by applying the original centralized algorithms on the entire network. This observation indicates that fast approximation algorithms, giving a good trade-off between high accuracy and low complexity, do not guarantee a good result in this situation. Indeed, LM finds the smallest number of the communities on the small partitions out of the three algorithms. By the same

token, GN is likely to reduce the error most rapidly as the partition radius increases.

### b: RUNNING TIME

In Figures 23(b) and 23(c), GN and CNM both tend to dramatically increase the running time gap between two consecutive radius steps along with partition radius. Compared to GN and CNM, LM shows a relatively small change in the gap. When the algorithm with high computational complexity is taken as a centralized algorithm, the running time is more profitable for our algorithm.

Taken together, the experimental results show that GN and CNM offer better conditions than LM for our algorithm. That is, they provide better approximations of modularity in a relatively short time. Furthermore, this brings a new advantage to our algorithm. The algorithms with high complexity, such as GN, are limited in terms of the size of networks they can adopt. PM enables them to analyze the networks considered too large to be tractable, as long as the networks are well distributed. As demonstrated in the previous section, PM actually allows them to get an adequate result, although it is still challenging to tackle large partitions.

## VII. CONCLUSION

In recent years, it has become increasingly important to design distributed high-performance graph computation

algorithms that can deal with large-scale networked data in a cloud-like distributed computation architecture. Inspired by this, in this paper, we have introduced Partition-Merge, a simple meta-algorithm, that takes an existing centralized algorithm and produces a distributed implementation. With the underlying graph having polynomial growth property, the resulting distributed implementation runs in essentially linear time and is as good as, and sometimes even better than the centralized algorithm. The experiments demonstrate the efficiency of the PM algorithm, finding that it actually produces a similar performance (better in some cases) to that of the centralized algorithm in a relatively short time.

The algorithm is applicable to any graph in general, and its computation time as well as performance guarantees depend on the underlying graph structure – interestingly enough, we have evaluated the performance guarantees for any graph. We strongly believe that such an algorithmic approach would be of great value for developing large-scale cloud-based graph computation facilities.

**APPENDIX  
PROOFS OF THEOREMS 1 AND 2**

**A. MAP INFERENCE**

In this Section, we first prove Theorem 1 and Theorem 2 for MAP inference.

*Bound on  $|E \setminus \cup_{k=1}^p E_k|$ :* We first state the following lemma which shows the essential property of the partition scheme. Lemma 1 and Lemma 2 both for MAP and modularity optimization. The proof of Lemma 1 is stated at the end of this section.

*Lemma 1:* Given  $G = (V, E)$  with polynomial growth of rate  $\rho \geq 1$  and associated constant  $C \geq 1$ , by choosing  $K = K(\rho, C, \delta)$  and  $\varepsilon = \varepsilon(\rho, C, \delta) = \frac{\delta}{4(2C-1)}$ , the partition scheme satisfies that for any edge  $e \in E$ ,

$$P(e \in \mathcal{B}) \leq 2\varepsilon. \tag{12}$$

*Lower bound on  $\mathcal{H}(\mathbf{x}^*)$ :* Here we provide a lower bound on  $\mathcal{H}^* = \mathcal{H}(\mathbf{x}^*)$  that will be useful to obtain multiplicative approximation property.

*Lemma 2:* Let  $\mathcal{H}^* = \max_{\mathbf{x} \in \Sigma^n} \mathcal{H}(\mathbf{x})$  denote the maximum value of  $\mathcal{H}$  for a given pair-wise MRF on a graph  $G$ . If  $G$  has maximum vertex degree  $d^*$ , then

$$(d^* + 1)\mathcal{H}(\mathbf{x}^*) \geq \sum_{(i,j) \in E} (\psi_{ij}^U - \psi_{ij}^L). \tag{13}$$

*Proof:* Assign weight  $w_{ij} = \psi_{ij}^U$  to an edge  $(i, j) \in E$ . Since graph  $G$  has maximum vertex degree  $d^*$ , by Vizing’s theorem there exists an edge-coloring of the graph using at most  $d^* + 1$  colors. Edges with the same color form a matching of the  $G$ . A standard application of Pigeon-hole’s principle implies that there is a color with weight at least  $\frac{1}{d^*+1}(\sum_{(i,j) \in E} w_{ij})$ . Let  $M \subset E$  denote these set of edges. Then

$$\sum_{(i,j) \in M} \psi_{ij}^U \geq \frac{1}{d^* + 1} \left( \sum_{(i,j) \in E} \psi_{ij}^U \right).$$

Now, consider an assignment  $\mathbf{x}^M$  as follows: for each  $(i, j) \in M$  set

$$(x_i^M, x_j^M) = \arg \max_{(x, x') \in \Sigma^2} \psi_{ij}(x, x'),$$

for remaining  $i \in V$ , set  $x_i^M$  to some value in  $\Sigma$  arbitrarily. Note that for above assignment to be possible, we have used matching property of  $M$ . Therefore, we have

$$\begin{aligned} \mathcal{H}(\mathbf{x}^M) &= \sum_{i \in V} \phi_i(x_i^M) + \sum_{(i,j) \in E} \psi_{ij}(x_i^M, x_j^M) \\ &= \sum_{i \in V} \phi_i(x_i^M) + \sum_{(i,j) \in E \setminus M} \psi_{ij}(x_i^M, x_j^M) \\ &\quad + \sum_{(i,j) \in M} \psi_{ij}(x_i^M, x_j^M) \\ &\stackrel{(a)}{\geq} \sum_{(i,j) \in M} \psi_{ij}(x_i^M, x_j^M) \\ &= \sum_{(i,j) \in M} \psi_{ij}^U \\ &\geq \frac{1}{d^* + 1} \left[ \sum_{(i,j) \in E} \psi_{ij}^U \right]. \end{aligned} \tag{14}$$

Here (a) follows because  $\psi_{ij}$  and  $\phi_i$  are non-negative valued functions. Since  $\mathcal{H}(\mathbf{x}^*) \geq \mathcal{H}(\mathbf{x}^M)$  and  $\psi_{ij}^L \geq 0$  for all  $(i, j) \in E$ , we prove Lemma 2.  $\square$

*Decomposition of  $\mathcal{H}^*$ :* Here we show that by maximizing  $\mathcal{H}(\cdot)$  on a partition of  $V$  separately and combining the assignments, the resulting  $\hat{\mathbf{x}}$  has  $\mathcal{H}(\cdot)$  value as good as that of MAP with penalty in terms of the edges across partitions.

*Lemma 3:* For a given MRF defined on  $G$ , the algorithm of the partition scheme produces output  $\hat{\mathbf{x}}$  such that

$$\mathcal{H}(\hat{\mathbf{x}}) \geq \mathcal{H}(\mathbf{x}^*) - \left( \sum_{(i,j) \in \mathcal{B}} (\psi_{ij}^U - \psi_{ij}^L) \right),$$

where  $\mathcal{B} = E \setminus \cup_{k=1}^K E_k$ ,  $\psi_{ij}^U \triangleq \max_{\sigma, \sigma' \in \Sigma} \ln \psi_{ij}(\sigma, \sigma')$ , and  $\psi_{ij}^L \triangleq \min_{\sigma, \sigma' \in \Sigma} \ln \psi_{ij}(\sigma, \sigma')$ .

*Proof:* Let  $\mathbf{x}^*$  be a MAP assignment of the MRF  $\mathbf{X}$  defined on  $G$ . Given an assignment  $\mathbf{x} \in \Sigma^{|V|}$  defined on a graph  $G = (V, E)$  and a subgraph  $S = (W, E')$  of  $G$ , let an assignment  $\mathbf{x}' \in \Sigma^{|W|}$  be called a restriction of  $\mathbf{x}$  to  $S$  if  $\mathbf{x}'(v) = \mathbf{x}(v)$  for all  $v \in W$ . Let  $S_1, \dots, S_K$  be the connected components of  $G' = (V, E - \mathcal{B})$ , and let  $\mathbf{x}_k^*$  be the restriction of  $\mathbf{x}^*$  to the component  $S_k$ . Let  $\mathbf{X}_k$  be the restriction of the MRF  $\mathbf{X}$  to  $G_k = (S_k, E_k)$ , where  $E_k = \{(u, w) \in E | u, w \in S_k\}$ .

For  $\mathbf{x}_k \in \Sigma^{|S_k|}$ , define

$$\mathcal{H}_k(\mathbf{x}_k) = \sum_{i \in S_k} \phi_i(x_i) + \sum_{(i,j) \in E_k} \psi_{ij}(x_i, x_j).$$

Let  $\hat{\mathbf{x}}$  be the output of the partition scheme, and let  $\hat{\mathbf{x}}_k$  be the restriction of  $\hat{\mathbf{x}}$  to the component  $S_k$ . Note that since  $\hat{\mathbf{x}}_k$  is a



MAP assignment of  $\mathcal{H}_k(\cdot)$  by the definition of our algorithm, for all  $k = 1, 2, \dots, K$ ,

$$\mathcal{H}_k(\widehat{\mathbf{x}}_k) \geq \mathcal{H}_k(\mathbf{x}_k^*). \quad (15)$$

Now, we have

$$\begin{aligned} \mathcal{H}(\widehat{\mathbf{x}}) - \mathcal{H}(\mathbf{x}^*) &= \sum_{k=1}^K [\mathcal{H}_k(\widehat{\mathbf{x}}_k) - \mathcal{H}_k(\mathbf{x}_k^*)] \\ &\quad + \sum_{(i,j) \in \mathcal{B}} \psi_{ij}(\widehat{x}_i, \widehat{x}_j) - \psi_{ij}(x_i^*, x_j^*) \\ &\stackrel{(a)}{\geq} \sum_{k=1}^K [\mathcal{H}_k(\widehat{\mathbf{x}}_k) - \mathcal{H}_k(\mathbf{x}_k^*)] \\ &\quad - \sum_{(i,j) \in \mathcal{B}} (\psi_{ij}^U - \psi_{ij}^L) \\ &\stackrel{(b)}{\geq} - \sum_{(i,j) \in \mathcal{B}} (\psi_{ij}^U - \psi_{ij}^L). \end{aligned} \quad (16)$$

Here (a) follows from the definitions of  $\psi_{ij}^U$  and  $\psi_{ij}^L$ , and (b) follows from (15). This completes the proof of Lemma 3.  $\square$

*Completing Proof of Theorem 1(a):* Recall that the maximum vertex degree  $d^*$  of  $G$  is less than  $2^\rho C$  by the definition of polynomially growing graph. Remind our definition  $\varepsilon = \frac{\delta}{2C2^\rho}$  for MAP inference. Now we have that

$$\mathbb{E}[\mathcal{H}(\widehat{\mathbf{x}})] \stackrel{(a)}{\geq} \mathcal{H}(\mathbf{x}^*) - \mathbb{E} \left[ \sum_{(i,j) \in \mathcal{B}} (\psi_{ij}^U - \psi_{ij}^L) \right] \quad (17)$$

$$\stackrel{(b)}{\geq} \mathcal{H}(\mathbf{x}^*) - 2\varepsilon \left( \sum_{(i,j) \in E} (\psi_{ij}^U - \psi_{ij}^L) \right) \quad (18)$$

$$\stackrel{(c)}{\geq} \mathcal{H}(\mathbf{x}^*) (1 - 2\varepsilon(d^* + 1)) \quad (19)$$

$$\stackrel{(d)}{\geq} (1 - \delta)\mathcal{H}(\mathbf{x}^*). \quad (20)$$

Here (a) follows from Lemma 3, (b) follows from Lemma 1, (c) from Lemma 2, and (d) follows from the definition of  $\varepsilon$  for MAP inference. This completes the proof of Theorem 1(a) for MAP inference.

*Completing Proof of Theorem 1(b):* Suppose that we use an approximation procedure  $\mathcal{A}$  to produce an approximate MAP assignment  $\widehat{\mathbf{x}}_k$  on each partition  $S_k$  in our algorithm. Let  $\mathcal{A}$  be such that the assignment produced satisfies that  $\mathcal{H}_k(\widehat{\mathbf{x}}_k)$  has value at least  $1/\alpha(n)$  times the maximum  $\mathcal{H}_k(\cdot)$  value for any graph of size  $n$ . Now since  $\mathcal{A}$  is applied to each partition separately, the approximation is within  $\alpha(\tilde{K})$  where  $\tilde{K} = CK^\rho$  is the bound on the number of nodes in each partition.

$$\mathcal{H}(\widehat{\mathbf{x}}_k) \geq \frac{1}{\alpha(\tilde{K})} \mathcal{M}(\mathbf{x}_k^*). \quad (21)$$

By the same proof of Lemma 3 together with (21), we have that

$$\mathbb{E}[\mathcal{H}(\widehat{\mathbf{x}})] \geq \frac{1}{\alpha(\tilde{K})} \mathcal{H}(\mathbf{x}^*) - \mathbb{E} \left[ \sum_{(i,j) \in \mathcal{B}} (\psi_{ij}^U - \psi_{ij}^L) \right]. \quad (22)$$

Hence we have that

$$\mathbb{E}[\mathcal{H}(\widehat{\mathbf{x}})] \geq \frac{1}{\alpha(\tilde{K})} \mathcal{H}(\mathbf{x}^*) - \mathbb{E} \left[ \sum_{(i,j) \in \mathcal{B}} (\psi_{ij}^U - \psi_{ij}^L) \right] \quad (23)$$

$$\stackrel{(a)}{\geq} \frac{1}{\alpha(\tilde{K})} \mathcal{H}(\mathbf{x}^*) - 2\varepsilon \left( \sum_{(i,j) \in E} (\psi_{ij}^U - \psi_{ij}^L) \right) \quad (24)$$

$$\stackrel{(b)}{\geq} \mathcal{H}(\mathbf{x}^*) \left( \frac{1}{\alpha(\tilde{K})} - 2\varepsilon(d^* + 1) \right) \quad (25)$$

$$\stackrel{(c)}{\geq} \left( \frac{1}{\alpha(\tilde{K})} - \delta \right) \mathcal{H}(\mathbf{x}^*). \quad (26)$$

Here (a) follows from Lemma 1, (b) follows from Lemma 2, and (c) from the definition of  $\varepsilon$  for MAP inference. This completes the proof of Theorem 1(b) for MAP inference.

*Completing Proof of Theorem 2:* The same arguments as in the proof Theorem 1 together with Lemma 3 completes the proof of Theorem 2 for MAP inference.

*Proof of Lemma 1:* Now we prove Lemma 1. First, we consider property of the partition scheme applied to a generic metric space  $\mathcal{G} = (V, \mathbf{d}_G)$ , where  $V$  is the set of points over which metric  $\mathbf{d}_G$  is defined. We state the result below for any metric space (rather than restricted to a graph) as it's necessary to carry out appropriate induction based proof. Note that the algorithm the partition scheme can be applied to any metric space (not just graph as well) as it only utilizes the property of metric in it's definition. The edge set  $E$  of metric space  $\mathcal{G}$  is precisely the set of all vertices that are within distance 1 of each other.

*Proposition 1:* Consider a metric space  $\mathcal{G} = (V, \mathbf{d}_G)$  defined over an  $n$  point set  $V$ , i.e.  $|V| = n$ . Let  $\mathcal{B} = E \setminus \cup_{k=1}^p E_k$  be the boundary set of the partition scheme applied to  $\mathcal{G}$ . Then, for any  $e \in E$ ,

$$\mathbb{P}[e \in \mathcal{B}] \leq \varepsilon + P_K \cdot |\mathbf{B}(e, K)|,$$

where  $\mathbf{B}(e, K) = \mathbf{B}_G(e, K)$  is the union of the two balls of radius  $K$  in  $\mathcal{G}$  with respect to the  $\mathbf{d}_G$  centered around the two end vertices of  $e$ , and  $P_K = (1 - \varepsilon)^{K-1}$ .

*Proof:* The proof is by induction on the number of points  $n$ . When  $n = 1$ , the algorithm chooses as the only point the point  $u_0$  in the initial iteration and hence no edge can be part of the output set  $\mathcal{B}$ . That is, for any edge, say  $e$ ,

$$\mathbb{P}[e \in \mathcal{B}] = 0 \leq \varepsilon + P_K |\mathbf{B}(e, K)|.$$

Thus, we have verified the base case for induction ( $n = 1$ ).

As induction hypothesis, suppose that the Proposition 1 is true for any graph with  $n$  nodes with  $n < N$  for some  $N \geq 2$ . As the induction step, we wish to establish

Proposition 1 for any  $\mathcal{G} = (V, \mathbf{d}_{\mathcal{G}})$  with  $|V| = N$ . For this, consider any  $v \in V$ . Now consider the last iteration of the the partition scheme applied to  $\mathcal{G}$ . The algorithm picks  $i_1 \in V$  uniformly at random in the first iteration. Given  $e$ , depending on the choice of  $i_1$  we consider three different cases (or events). We will show that in these three cases,

$$\mathbb{P}[e \in \mathcal{B}] \leq \varepsilon + P_K |\mathbf{B}(e, K)|$$

holds.

*Case 1.* Suppose  $i_1$  is such that  $\mathbf{d}_{\mathcal{G}}(i_1, e) < K$ , where the distance of a point and an edge of  $\mathcal{G}$  is defined as a minimum distance from the point to one of the two end-points of the edge. Call this event  $E_1$ . Further, depending on choice of random number  $R_1$ , define the following events

$$\begin{aligned} E_{11} &= \{\mathbf{d}_{\mathcal{G}}(i_1, e) < R_1\}, E_{12} = \{\mathbf{d}_{\mathcal{G}}(i_1, e) = R_1\}, \\ &\text{and } E_{13} = \{\mathbf{d}_{\mathcal{G}}(i_1, e) > R_1\}. \end{aligned}$$

By the definition of the partition scheme, when  $E_{11}$  happens,  $e$  can never be a part of  $\mathcal{B}$ . When  $E_{12}$  happens,  $e$  is definitely a part of  $\mathcal{B}$ . When  $E_{13}$  happens, it is said to be left as an element of the set  $\mathcal{W}_1$ . This new vertex set  $\mathcal{W}_1$  has points less than  $N$ . The original metric  $\mathbf{d}_{\mathcal{G}}$  is still considered as the metric on the points<sup>5</sup> of  $\mathcal{W}_1$ . By its definition, the partition scheme excluding the first iteration is the same as the partition scheme applied to  $(\mathcal{W}_1, \mathbf{d}_{\mathcal{G}})$ . Therefore, we can invoke induction hypothesis which implies that if event  $E_{13}$  happens then the probability of  $v \in \mathcal{B}$  is bounded above by  $\varepsilon + P_K \cdot |\mathbf{B}(e, K)|$ , where  $\mathbf{B}(e, K)$  is the ball with respect to  $(\mathcal{W}_1, \mathbf{d}_{\mathcal{G}})$  which has no more than the number of points in the ball  $\mathbf{B}(e, K)$  defined with respect to the original metric space  $\mathcal{G}$ . Finally, let us relate the  $\mathbb{P}[E_{11}|E_2]$  with  $\mathbb{P}[E_{12}|E_1]$ . Suppose  $\mathbf{d}_{\mathcal{G}}(i_1, e) = \ell < K$ . By the definition of probability distribution of  $\mathbf{Q}$ , we have

$$\mathbb{P}[E_{12}|E_1] = \varepsilon(1 - \varepsilon)^{\ell-1}, \quad (27)$$

$$\begin{aligned} rCI\mathbb{P}[E_{11}|E_1] &= (1 - \varepsilon)^{K-1} + \sum_{j=\ell+1}^{K-1} \varepsilon(1 - \varepsilon)^{j-1} \\ &= (1 - \varepsilon)^{\ell}. \end{aligned} \quad (28)$$

That is,

$$\mathbb{P}[E_{12}|E_1] = \frac{\varepsilon}{1 - \varepsilon} \mathbb{P}[E_{11}|E_1].$$

Let  $q \triangleq \mathbb{P}[E_{11}|E_1]$ . Then,

$$\begin{aligned} \mathbb{P}[e \in \mathcal{B}|E_1] &= \mathbb{P}[e \in \mathcal{B}|E_{11} \cap E_1] \mathbb{P}[E_{11}|E_1] \\ &\quad + \mathbb{P}[e \in \mathcal{B}|E_{12} \cap E_1] \mathbb{P}[E_{12}|E_1] \\ &\quad + \mathbb{P}[e \in \mathcal{B}|E_{13} \cap E_1] \mathbb{P}[E_{13}|E_1] \\ &\leq 0 \times q + 1 \times \frac{\varepsilon q}{1 - \varepsilon} \\ &\quad + (\varepsilon + P_K |\mathbf{B}(e, K)|) \left(1 - \frac{q}{1 - \varepsilon}\right) \end{aligned}$$

<sup>5</sup>Note the following subtle but crucial point. We are not changing the metric  $\mathbf{d}_{\mathcal{G}}$  after we remove points from the original set of points.

$$\begin{aligned} &= \varepsilon + P_K |\mathbf{B}(e, K)| \\ &\quad + \frac{q}{1 - \varepsilon} (\varepsilon - \varepsilon - P_K |\mathbf{B}(e, K)|) \\ &= \varepsilon + P_K |\mathbf{B}(e, K)| - \frac{q P_K |\mathbf{B}(e, K)|}{1 - \varepsilon} \\ &\leq \varepsilon + P_K |\mathbf{B}(e, K)|. \end{aligned} \quad (29)$$

*Case 2.* Now, suppose  $i_1 \in V$  is such that  $\mathbf{d}_{\mathcal{G}}(i_1, e) = K$ . We will call this event  $E_2$ . Further, define the event  $E_{21} = \{R_1 = K\}$ . Due to the independence of selection of  $R_1$ ,  $\mathbb{P}[E_{21}|E_2] = P_K$ . Under the event  $E_{21} \cap E_2$ ,  $e \in \mathcal{B}$  with probability 1. Therefore,

$$\begin{aligned} \mathbb{P}[e \in \mathcal{B}|E_2] &= \mathbb{P}[e \in \mathcal{B}|E_{21} \cap E_2] \mathbb{P}[E_{21}|E_2] \\ &\quad + \mathbb{P}[e \in \mathcal{B}|E_{21}^c \cap E_2] \mathbb{P}[E_{21}^c|E_2] \\ &= 1 \times P_K + \mathbb{P}[e \in \mathcal{B}|E_{21}^c \cap E_2] (1 - P_K). \end{aligned} \quad (30)$$

Under the event  $E_{21}^c \cap E_2$ , we have  $e \in \mathcal{W}_1$ , and the remaining metric space  $(\mathcal{W}_1, \mathbf{d}_{\mathcal{G}})$ . This metric space has  $< N$  points. Further, the ball of radius  $K$  around  $e$  with respect to this new metric space has at most  $|\mathbf{B}(e, K)| - 1$  points (this ball is with respect to the original metric space  $\mathcal{G}$  on  $N$  points). Now we can invoke the induction hypothesis for this new metric space to obtain

$$\mathbb{P}[e \in \mathcal{B}|E_{21}^c \cap E_2] \leq \varepsilon + P_K \cdot (|\mathbf{B}(e, K)| - 1). \quad (31)$$

From (30) and (31), we have

$$\begin{aligned} \mathbb{P}[e \in \mathcal{B}|E_2] &\leq P_K + (1 - P_K)(\varepsilon + P_K \cdot (|\mathbf{B}(e, K)| - 1)) \\ &= \varepsilon(1 - P_K) + P_K |\mathbf{B}(e, K)| \\ &\quad + P_K^2 (1 - |\mathbf{B}(e, K)|) \\ &\leq \varepsilon + P_K |\mathbf{B}(e, K)|. \end{aligned}$$

In above, we have used the fact that  $|\mathbf{B}(e, K)| \geq 1$  (or else, the bound was trivial to begin with).

*Case 3.* Finally, let  $E_3$  be the event that  $\mathbf{d}_{\mathcal{G}}(i_1, e) > K$ . Then, at the end of the first iteration of the algorithm, we again have the remaining metric space  $(\mathcal{W}_1, \mathbf{d}_{\mathcal{G}})$  such that  $|\mathcal{W}_1| < N$ . Hence, as before, by induction hypothesis we have

$$\mathbb{P}[e \in \mathcal{B}|E_3] \leq \varepsilon + P_K |\mathbf{B}(e, K)|.$$

Now, the three cases are exhaustive and disjoint. That is,  $\cup_{i=1}^3 E_i$  is the universe. Based on the above discussion, we obtain the following.

$$\begin{aligned} \mathbb{P}[e \in \mathcal{B}] &= \sum_{i=1}^3 \mathbb{P}[e \in \mathcal{B}|E_i] \mathbb{P}[E_i] \\ &\leq \left(\max_{i=1}^3 \mathbb{P}[e \in \mathcal{B}|E_i]\right) \left(\sum_{i=1}^3 \mathbb{P}[E_i]\right) \\ &\leq \varepsilon + P_K \cdot |\mathbf{B}(e, K)|. \end{aligned} \quad (32)$$

This completes the proof of Proposition 1.  $\square$

Now, we will use Proposition 1 to complete the proof of Lemma 1. The definition of growth rate implies that,

$$|\mathbf{B}(e, K)| \leq C \cdot K^\rho.$$

From the definition  $P_K = (1 - \varepsilon)^{K-1}$ , we have

$$P_K |\mathbf{B}(e, K)| \leq C(1 - \varepsilon)^{K-1} K^\rho.$$

Therefore, to show Lemma 1, it is sufficient to show that our definition of  $K$  satisfies the following Lemma.

*Lemma 4: We have that*

$$C(1 - \varepsilon)^{K-1} K^\rho \leq \varepsilon.$$

*Proof:* We will show the following equivalent inequality.

$$(K - 1) \log(1 - \varepsilon)^{-1} \geq \rho \log K + \log C + \log \frac{1}{\varepsilon}. \quad (33)$$

First, note that for all  $\varepsilon \in (0, 1)$ ,

$$\log(1 - \varepsilon)^{-1} \geq \log(1 + \varepsilon) \geq \frac{\varepsilon}{2}.$$

Hence to prove (33), it is sufficient to show that

$$K \geq \frac{2\rho}{\varepsilon} \log K + \frac{2}{\varepsilon} \log C + \frac{2}{\varepsilon} \log \frac{1}{\varepsilon} + 1. \quad (34)$$

Recall that

$$K = K(\varepsilon, \rho) = \frac{8\rho}{\varepsilon} \log \left( \frac{8\rho}{\varepsilon} \right) + \frac{4}{\varepsilon} \log C + \frac{4}{\varepsilon} \log \frac{1}{\varepsilon} + 2.$$

From the definition of  $K$ , we will show that

$$\frac{K}{2} \geq \frac{2\rho}{\varepsilon} \log K$$

and

$$\frac{K}{2} \geq \frac{2}{\varepsilon} \log C + \frac{2}{\varepsilon} \log \frac{1}{\varepsilon} + 1,$$

which will prove (34). The following is straightforward:

$$\frac{K}{2} \geq \frac{2}{\varepsilon} \log C + \frac{2}{\varepsilon} \log \frac{1}{\varepsilon} + 1. \quad (35)$$

Now, let  $\widehat{K} = \frac{8\rho}{\varepsilon} \log \left( \frac{8\rho}{\varepsilon} \right)$ . Then

$$\begin{aligned} \frac{\widehat{K}}{2} &= \frac{4\rho}{\varepsilon} \log \left( \frac{8\rho}{\varepsilon} \right) \\ &\geq \frac{2\rho}{\varepsilon} \left( \log \left( \frac{8\rho}{\varepsilon} \right) + \log \log \left( \frac{8\rho}{\varepsilon} \right) \right) \\ &= \frac{2\rho}{\varepsilon} \log \widehat{K}. \end{aligned}$$

That is,  $\frac{\widehat{K}}{2} - \frac{2\rho}{\varepsilon} \log \widehat{K} \geq 0$ . Since the function  $\phi(x) = \frac{x}{2} - \frac{2\rho}{\varepsilon} \log x$  is an increasing function of  $x$  when  $x \geq \frac{4\rho}{\varepsilon}$ , and from the fact that  $K \geq \widehat{K} \geq \frac{4\rho}{\varepsilon}$ , we have

$$\frac{K}{2} \geq \frac{2\rho}{\varepsilon} \log K. \quad (36)$$

From (35) and (36), we have (34), which completes the proof of Lemma 4.  $\square$

## B. MODULARITY OPTIMIZATION

In this Section, we prove Theorem 1 and Theorem 2 for modularity optimization.

*Lower bound on  $\mathcal{M}^*$ :* Here we provide a lower bound on  $\mathcal{M}^*$  that will be useful to obtain multiplicative approximation property.

*Lemma 5: Let  $\mathcal{M}^* = \max_{\chi} \mathcal{M}(\chi)$  denote the maximum value of modularity for graph  $G$ . Then,*

$$\mathcal{M}^* \geq \frac{1}{2(2C - 1)} \left( 1 - \frac{C^2}{2m} \right).$$

*Proof:* Since the graph has polynomial growth with degree  $\rho$  and associated constant  $C$ , it follows that the number of nodes within one hop of any node  $i \in V$  (i.e. its immediate neighbors) is at most  $C$ . That is,  $d_i \leq C$  for all  $i \in V$ . Given this bound, it follows that there exists a matching of size at least  $m/(2C - 1)$  in  $G$ . Given such a matching, consider the following clustering (coloring). Each edge in the matching represent a community of size 2, while all the nodes that are unmatched lead to community of size 1. By definition, the individual (unmatched) nodes contribute 0 to the modularity. The nodes that are part of the two node communities, each contribute at least  $\frac{1}{2m} \left( 1 - \frac{C^2}{2m} \right)$  since vertex degree of each node is bounded above by  $C$ . Since there are  $m/(2C - 1)$  edges in the matching, it follows that the net modularity of such community assignment is at least  $\frac{1}{2(2C-1)} \left( 1 - \frac{C^2}{2m} \right)$ . This completes the proof of Lemma 5 (Similar result, with tighter constant, follows from [35]).  $\square$

*Decomposition of  $\mathcal{M}^*$ :* Here we show that by maximizing modularity on a partition of  $V$  separately, the resulting clustering has modularity as good as that of optimal partitioning with penalty in terms of the edges across partitions. To that end, let  $V = V_1 \cup \dots \cup V_p$  be a partition of  $V$ , i.e.  $V_i \cap V_j = \emptyset$  for  $i \neq j$ . Let  $G_k = (V_k, E_k)$ , where  $E_k = (V_k \times V_k) \cap E$ , denote the subgraph of  $G$  for  $1 \leq k \leq p$ . Let  $\chi^k$  be a coloring (clustering) of  $G_k$  with maximum modularity. Let  $\chi^*$  be a coloring of  $G$  with maximum modularity ( $\mathcal{M}^*$ ) and let  $\chi^{*,k}$  be the restriction of  $\chi^*$  to  $G_k$ . Let  $\widehat{\chi}$  denote the clustering of  $G$  obtained by taking union of clusterings  $\chi^1, \dots, \chi^p$ . Then we claim the following.

*Lemma 6: For any partition  $V = V_1 \cup \dots \cup V_p$ ,*

$$\mathcal{M}(\widehat{\chi}) \geq \mathcal{M}(\chi^*) - \frac{1}{2m} |E \setminus \cup_{k=1}^p E_k|.$$

*Proof:* Consider the following:

$$\begin{aligned} 2m \mathcal{M}(\widehat{\chi}) &= \sum_{i,j \in V} \mathbf{1}_{\{\widehat{\chi}(i)=\widehat{\chi}(j)\}} \left( A_{ij} - \frac{d_i d_j}{2m} \right) \\ &\stackrel{(a)}{=} \sum_{k=1}^p \sum_{i,j \in V_k} \mathbf{1}_{\{\chi^k(i)=\chi^k(j)\}} \left( A_{ij} - \frac{d_i d_j}{2m} \right) \\ &\stackrel{(b)}{\geq} \sum_{k=1}^p \sum_{i,j \in V_k} \mathbf{1}_{\{\chi^{*,k}(i)=\chi^{*,k}(j)\}} \left( A_{ij} - \frac{d_i d_j}{2m} \right) \\ &= \sum_{i,j \in V} \mathbf{1}_{\{\chi^*(i)=\chi^*(j)\}} \left( A_{ij} - \frac{d_i d_j}{2m} \right) \end{aligned}$$

$$- \sum_{(i,j) \in V^2 \setminus \cup_{k=1}^p V_k^2} \mathbf{1}_{\{\chi^*(i)=\chi^*(j)\}} \left( A_{ij} - \frac{d_i d_j}{2m} \right) \tag{37}$$

$$\geq \sum_{i,j \in V} \mathbf{1}_{\{\chi^*(i)=\chi^*(j)\}} \left( A_{ij} - \frac{d_i d_j}{2m} \right) - |E \setminus \cup_{k=1}^p E_k|, \tag{38}$$

where the last inequality follows because the term inside the summation in (37) is positive only if  $A_{ij} = 1$ , i.e.  $(i, j) \in E$  or else it is negative. Therefore, for the purpose of lower bound, we only need to worry about  $(i, j) \in E$  such that  $(i, j) \notin \cup_{k=1}^p V_k \times V_k$ . This is precisely equal to  $E \setminus \cup_{k=1}^p E_k$ . The (a) follows because  $\widehat{\chi}$ , by definition, assigns nodes in  $V^i$  and  $V^j$  for  $i \neq j$  to different clusters. The (b) follows because  $\chi^k$  has maximum modularity in  $G_k$  and hence it is at least as large (in terms of modularity) as that of the  $\chi^{*,k}$ , the restriction of  $\chi^*$  to  $G_k$ . This completes the proof of Lemma 6 since the first term in (38) is precisely  $2m\mathcal{M}(\chi^*) = 2m\mathcal{M}^*$ .  $\square$

*Approximation Factor for  $\mathcal{M}(\widehat{\chi})$ :* Let  $\beta = |E \setminus \cup_{k=1}^p E_k|/m$  denote the fraction of edges that are across partitions for a given partition  $V = V_1 \cup \dots \cup V_p$ . Then, from Lemmas 5 and 6, it follows that for  $m \geq C^2$ ,

$$\begin{aligned} \mathcal{M}(\widehat{\chi}) &\geq \mathcal{M}(\chi^*) \left( 1 - \frac{\beta}{2\mathcal{M}(\chi^*)} \right) \\ &\geq \mathcal{M}(\chi^*) (1 - 2(2C - 1)\beta). \end{aligned} \tag{39}$$

Therefore, if  $2(2C - 1)\beta \leq \delta$ , then  $\mathcal{M}(\widehat{\chi})$  is at least  $\mathcal{M}^* \cdot (1 - \delta)$ . Now from Lemma 1 and the linearity of expectation, we have

$$\mathbb{E}[|E \setminus \cup_{k=1}^p E_k|] \leq \frac{\delta}{2(2C - 1)} m. \tag{40}$$

*Completing Proof of Theorem 1(a):* When  $\mathcal{A}$  produces exact solution to the modularity optimization for each partition, the resulting solution of our algorithm is  $\widehat{\chi}$ . Therefore, from (39) and (40), it follows that

$$\mathbb{E}[\mathcal{M}(\widehat{\chi})] \geq \mathcal{M}(\chi^*) (1 - \delta). \tag{41}$$

*Completing Proof of Theorem 1(b):* Suppose we use an approximation procedure  $\mathcal{A}$  to produce clustering on each partition in our algorithm. Let  $\mathcal{A}$  be such that the clustering produced has modularity at least  $1/\alpha(n)$  times the optimal modularity for any graph of size  $n$ . Now since  $\mathcal{A}$  is applied to each partition separately, the approximation is within  $\alpha(\tilde{K})$  where  $\tilde{K} = CK^\rho$  is the bound on the number of nodes in each partition. Let  $\tilde{\chi}^1, \dots, \tilde{\chi}^p$  be the clustering (coloring) produced by  $\mathcal{A}$  on graphs  $G_1, \dots, G_p$ . Then by the approximation property of  $\mathcal{A}$ , we have

$$\mathcal{M}(\tilde{\chi}^k) \geq \frac{1}{\alpha(\tilde{K})} \mathcal{M}(\chi^k). \tag{42}$$

Therefore, for the overall clustering  $\tilde{\chi}$  obtained as union of  $\tilde{\chi}^1, \dots, \tilde{\chi}^p$ , we have

$$\mathcal{M}(\tilde{\chi}) = \sum_{k=1}^p \mathcal{M}(\tilde{\chi}^k)$$

$$\begin{aligned} &\geq \frac{1}{\alpha(\tilde{K})} \sum_{k=1}^p \mathcal{M}(\chi^k) \\ &= \frac{1}{\alpha(\tilde{K})} \mathcal{M}(\widehat{\chi}). \end{aligned} \tag{43}$$

Since  $\mathbb{E}[\mathcal{M}(\widehat{\chi})]$  is at least  $(1 - \delta)\mathcal{M}^*$ , it follows that  $\mathbb{E}[\mathcal{M}(\tilde{\chi})] \geq \frac{(1-\delta)}{\alpha(\tilde{K})} \mathcal{M}^*$ .

*Completing Proof of Theorem 2:* Lemma 6 directly proves Theorem 2(a), and the same arguments as in the proof Theorem 1(b) completes the proof of Theorem 2(b).

### ACKNOWLEDGMENT

Part of this work appeared in the preliminary version [1].

### REFERENCES

- [1] K. Jung, P. Kohli, and D. Shah, "Local rules for global map: When do they work?" in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 22, 2009, pp. 871–879.
- [2] S. Sarkhel, P. Singla, and V. G. Gogate, "Fast lifted map inference via partitioning," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 3240–3248.
- [3] W. Yang, G. Wang, M. Z. A. Bhuiyan, and K.-K.-R. Choo, "Hypergraph partitioning for social networks based on information entropy modularity," *J. Netw. Comput. Appl.*, vol. 86, pp. 59–71, May 2017.
- [4] D. A. Spielman and S.-H. Teng, "A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning," *SIAM J. Comput.*, vol. 42, no. 1, pp. 1–26, Jan. 2013.
- [5] W. Fan, R. Jin, M. Liu, P. Lu, X. Luo, R. Xu, Q. Yin, W. Yu, and J. Zhou, "Application driven graph partitioning," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2020, pp. 1765–1779.
- [6] I. Stanton, "Streaming balanced graph partitioning algorithms for random graphs," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2014, pp. 1287–1301.
- [7] I. Stanton and G. Kliot, "Streaming graph partitioning for large distributed graphs," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1222–1230.
- [8] J. Nishimura and J. Ugander, "Restreaming graph partitioning: Simple versatile algorithms for advanced balancing," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2013, pp. 1106–1114.
- [9] W. Fan, M. Liu, C. Tian, R. Xu, and J. Zhou, "Incrementalization of graph partitioning algorithms," *Proc. VLDB Endowment*, vol. 13, no. 8, pp. 1261–1274, Apr. 2020.
- [10] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1568–1583, Oct. 2006.
- [11] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, Apr. 2002.
- [12] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 70, Dec. 2004, Art. no. 066111.
- [13] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.
- [14] B. Awerbuch, M. Luby, A. V. Goldberg, and S. A. Plotkin, "Network decomposition and locality in distributed computation," in *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 1989, pp. 364–369.
- [15] P. Klein, S. A. Plotkin, and S. Rao, "Excluded minors, network decomposition, and multicommodity flow," in *Proc. 25th Annu. ACM Symp. Theory Comput. (STOC)*, 1993, pp. 682–690.
- [16] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, vol. 5. Philadelphia, PA, USA: Society for Industrial Mathematics, 2000.
- [17] A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak, "Local graph partitions for approximation and testing," in *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2009, pp. 22–31.
- [18] K. Jung and D. Shah, "Local algorithms for approximate inference in minor-excluded graphs," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, 2007, pp. 1–12.



- [19] S. K. Andersen, "Probabilistic reasoning in intelligent systems: Networks of plausible inference," *Artif. Intell.*, vol. 48, no. 1, pp. 117–124, Feb. 1991.
- [20] J. Yedidia, W. Freeman, and Y. Weiss, "Generalized belief propagation," Mitsubishi Elect. Res. Lab., Cambridge, MA, USA, Tech. Rep. TR-2000-26, 2000.
- [21] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "MAP estimation via agreement on trees: Message-passing and linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 11, pp. 3697–3717, Nov. 2005.
- [22] M. Bayati, D. Shah, and M. Sharma, "Maximum weight matching via max-product belief propagation," in *Proc. IEEE ISIT*, Sep. 2005, pp. 1763–1767.
- [23] M. Bayati, D. Shah, and M. Sharma, "Max-product for maximum weight matching: Convergence, correctness, and LP duality," *IEEE Trans. Inf. Theory*, vol. 54, no. 3, pp. 1241–1251, Mar. 2008.
- [24] B. Huang and T. Jebara, "Loopy belief propagation for bipartite maximum weight B-matching," in *Proc. Artif. Intell. Statist. (AISTATS)*, 2007, pp. 195–202.
- [25] S. Sanghavi, D. Shah, and A. Willsky, "Message passing for maximum weight independent set," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2007, pp. 4822–4834.
- [26] D. Sontag and T. Jaakkola, "Tree block coordinate descent for map in graphical models," *J. Mach. Learn. Res.-Proc. Track*, vol. 5, pp. 544–551, Dec. 2009.
- [27] D. L. Tarlow, D. Batra, P. Kohli, and V. Kolmogorov, "Dynamic tree block coordinate ascent," in *Proc. ICMML*, 2011, pp. 1–8.
- [28] R. H. Swendsen and J.-S. Wang, "Nonuniversal critical dynamics in Monte Carlo simulations," *Phys. Rev. Lett.*, vol. 58, no. 2, pp. 86–88, Jan. 1987.
- [29] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, p. 8577, 2006.
- [30] V. Blondel, G. Krings, and I. Thomas, "Regions and borders of mobile telephony in Belgium and in the Brussels metropolitan zone," *Brussels Stud.*, vol. 42, no. 4, Oct. 2010. [Online]. Available: <http://journals.openedition.org/brussels/806>, doi: [10.4000/brussels.806](https://doi.org/10.4000/brussels.806).
- [31] M. E. J. Newman, "Community detection and graph partitioning," *Europhys. Lett.*, vol. 103, no. 2, p. 28003, Jul. 2013.
- [32] B. DasGupta and D. Desai, "On the complexity of Newman's community finding approach for biological and social networks," 2011, *arXiv:1102.0969*. [Online]. Available: <http://arxiv.org/abs/1102.0969>
- [33] A. Gupta, R. Krauthgamer, and J. R. Lee, "Bounded geometries, fractals, and low-distortion embeddings," in *Proc. Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2003, pp. 534–543.
- [34] R. Gummadi, K. Jung, D. Shah, and R. Sreenivas, "Computing the capacity region of a wireless network," in *Proc. 28th Conf. Comput. Commun.*, Apr. 2009, pp. 1341–1349.
- [35] Y. Han, "Matching for graphs of bounded degree," in *Proc. Int. Workshop Frontiers Algorithmics*, 2008, pp. 171–173.



**VINCENT BLONDEL** received the degree in engineering and the degree in philosophy from the Université catholique de Louvain (UCL), Belgium, the M.Sc. degree in pure mathematics from Imperial College, London, U.K., and the Ph.D. degree in applied mathematics from UCL. He is currently a Professor of applied mathematics and the President of UCL. His major research interests include mathematical control theory, theoretical computer science, and network science. He has also completed a master thesis at the Institut National Polytechnique de Grenoble, France. For his scientific contributions, he has been awarded the triennial SIAM prize on Control and Systems Theory, in 2001, the prize Adolphe Wetrem of the Belgian Royal Academy of Science, in 2006, and the Antonio Ruberti prize in Systems and Control of the IEEE, in 2006. His recent work on networks has been widely featured, including in *Wired*, *Technology Review*, *Le Monde*, *La Recherche*, *Der Spiegel*, *The Wall Street Journal*, and *The New York Times*.



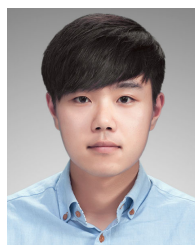
**KYOMIN JUNG** (Member, IEEE) received the B.S. degree in mathematics from Seoul National University (SNU), Seoul, South Korea, in 2003, and the Ph.D. degree in mathematics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2009. From 2009 to 2013, he was an Assistant Professor with the Department of Computer Science, KAIST. Since 2016, he has been an Assistant Professor and an Associate Professor with the Department of Electrical and Computer Engineering, SNU. He is currently an Adjunct Professor with the Department of Mathematical Sciences, SNU. His research interests include natural language processing, deep learning and applications, data analysis, and web services.



**PUSHMEET KOHLI** is currently the Head of Research for Science, Robustness, and Reliability with DeepMind. Before joining DeepMind, he was the Director of Research with the Cognition Group, Microsoft. During his ten years at Microsoft, he worked with Microsoft Labs in Seattle, Cambridge, and Bengaluru, and enacted several roles and responsibilities; including being a Technical Advisor to Rick Rashid, the Chief Research Officer of Microsoft. He published in the fields of machine learning, computer vision, information retrieval, and game theory. His papers have appeared in computer vision, machine learning, robotics, and AI conferences. His research interests include intelligent systems and computational sciences. He has received awards in ICVGIP 2006, 2010, ECCV 2010, ISMAR 2011, TVX 2014, CHI 2014, WWW 2014, and CVPR 2015. His research has also been the subject of a number of articles in popular media outlets, such as *Forbes*, *Wired*, *BBC*, *New Scientist*, and *MIT Technology Review*.



**DEVAVRAT SHAH** (Senior Member, IEEE) received the B.Tech. degree from IIT Bombay, in 1999, and the Ph.D. degree from Stanford, in 2004. He is currently a Professor with the Department of Electrical Engineering and Computer Science, MIT. He is also the Director of the Statistics and Data Science Center, Institute for Data, Systems and Society. He founded machine learning start-up Celect, in 2013, which is currently part of Nike, since 2019. His research has primarily focused on developing practical algorithmic solutions with theoretical guarantees for machine learning and statistical inference, social data processing, and communication networks, including Internet, wireless, and data centers. His work has been recognized through best publication awards at *NeurIPS*, *ACM Sigmetrics*, *IEEE Infocom*, *INFORMS Applied Probability Society*, *INFORMS Operations Management*, *INFORMS Revenue Management and Pricing*, and *INFORMS Marketing Science*. He has received the 2010 Erlang Prize for *INFORMS Applied Probability Society* and 2008 *ACM Sigmetrics Rising Star Award*, and the Test of Time Paper Awards at *ACM Sigmetrics 2019* and 2020. He received the President of India Gold Medal for his B.Tech. degree.



**SEUNGPIIL WON** received the B.S. degree in electrical and electronic engineering from Yonsei University, South Korea, in 2015. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Seoul National University, South Korea. His research interests focus on the areas that have benefitted from generative models, including natural language processing and computer vision.

...