Partition-Merge: Distributed Inference and Modularity Optimization

Vincent Blondel VINCENT.BLONDEL@UCLOUVAIN.BE
Universite Catholique de Louvain, Belgium

Kyomin Jung KJUNG@SNU.AC.KR
Seoul National University, South Korea

Pushmeet Kohli PUSHMEET@GOOGLE.COM
DeepMind, London, UK

Devavrat Shah DEVAVRAT@MIT.EDU
MIT, Cambridge, USA

Seungpil Won SEUNGPIL.WON@SNU.AC.KR
Seoul National University, South Korea

Editor: Editor

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 essentially run in time linear in the number of nodes in the graph, and perform as well – or even better – than the original centralized algorithm as long as the graph has geometric structures[1]. More precisely, if the centralized algorithm is a $\mathcal{C}-$factor approximation with constant $\mathcal{C} \geq 1$, the resulting distributed algorithm is a $(\mathcal{C}+\delta)$-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 efficiency of our algorithm, we conducted extensive experiments both on real-world networks and on synthetic networks. For the centralized algorithms, we consider the sequential tree-reweighted max-product message passing for the MAP inference, and Girvan-Newman, Clauset-Newman-Moore, and Louvain-Method for the modularity optimization problem. The experiments demonstrate that PM provides a good trade-off between accuracy and running time. It particularly achieves better efficiency when it is applied to well-distributed regular networks and when the centralized algorithm has high complexity.

Keywords: Graphical model, Approximate MAP, Modularity Optimization, Partition

[1] 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.

1. 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 and remain very active areas of current investigation.

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 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, δ is a small number that depends on a tuneable parameter of the algorithm that affects the size of the induced subgraphs in the partition; the larger the subgraph size, the smaller the δ. 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 its performance.

The algorithm applies to any graph structure, but strong guarantees on performance, as stated above, require geometric structure. 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 4.2.

A cautionary remark is in order. Indeed, by no means, this algorithm means to answer all problems in distributed computation. Specifically, for dense graph, 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 these problem therefore require differen t way to think about them.

In order to verify the validity of the PM algorithm, we have applied it both on real-world networks and on synthetic graphs, comparing it with the original centralized algorithms. For MAP inference, we used the sequential tree-reweighted max-product message passing (TRW-S) Kolmogorov (2006) with the energy function defined in Section 7.2. For modularity optimization, we selected three different algorithms from old-fashioned way to state-of-the-art: Girvan-Newman (GN) Girvan and Newman (2002), Clauset-Newman-Moore (CNM) Clauset et al. (2004), and Louvain-Method (LM) Blondel et al. (2008). 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 case of a comparatively dense network, such as Barabasi-Albert

model, PM produces a decent result although the efficiency of PM is relatively low. Furthermore, PM particularly performs better when applied on well-distributed regular networks and when the centralized algorithm has high complexity. In addition, 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 nodes in the network.

Related Work and Our contributions. The results of this paper, on 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., Awerbuch et al. (1989); Klein et al. (1993) and Peleg (2000) for a detailed account of the approach until 2000. More recently, such decompositions have found wide variety of applications including local-property testing Hassidim et al. (2009). All such decompositions are useful for *homogeneous* problems, e.g. for finding maximum-*size* matching or independent set rather than the *heterogenous* maximum-*weight* variants of it. To overcome this limitation, a different (somewhat stronger) notion of decomposition was introduced by Jung and Shah Jung and Shah (2007) for minor-excluded graphs that built upon Klein et al. (1993). 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 in $1/\epsilon$. 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 by 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 graph with polynomial growth. Then the resulting distributed algorithm becomes very fast in practice, unlike the dynamic programming approach, 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 earlier, 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 Jung and Shah (2007).

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 instance of NP-complete problems.

MAP Inference. Computing the exact Maximum a Posteriori (MAP) solution in a general probabilistic model is an NP-hard problem. A number of 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. Keeping the values of all other variables fixed, the value of the selected variable is chosen 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 are related to max-product belief propagation (cf. Pearl (1988) and Yedidia et al. (2000)). In recent years a sequence of results suggest that there is an intimate relation between the max-product algorithm and a natural linear programming relaxation – for example, see Wainwright et al. (2005); Bayati et al. (2005, 2008); Huang and Jebara (2007); Sanghavi et al. (2007). Many of these methods can be seen as making local updates to partitions of the dual problem Sontag and Jaakkola (2009); Tarlow et al. (2011).

We also note that Swendsen-Wang algorithm (SW)Swendsen and Wang (1987), 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 local update based algorithm are primarily for linear or more generally convex optimization setup.

Modularity Optimization for Clustering. The notion of modularity optimization was introduced by Newmann Newman (2006) 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 variety of networked data cf. Blondel et al. (2008, 2010). The major challenge has been design of approximation algorithm for modularity optimization (which is computationally hard in general) that can operate in distributed manner and provide performance guarantees. Such algorithms with provable performance guarantees are known only for few cases, notably logarithmic approximation of DasGupta and Desai (2011) 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 an distributed and fast approximation algorithm through our approach.

Organization. The rest of the paper is organized as follows. Section 2 describes the problem statement and preliminaries. Section 3 describes our main algorithms, and Section 4 presents analyses of our algorithms. Section A.1 and Section A.2 provide the proofs of our main theorems. Section 5 and Section 6 present the setup and results of an experiment, respectively, and Section 7 presents the conclusion.

2. Setup

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 $\mathbf{d_G} : V \times V \to \mathbb{R}_+$ with $\mathbf{d_G}(i, j)$ given by the length of the shortest path between i and j; defined as ∞ 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) ρ, 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 $\mathbf{B}_G(i, r) = \{j \in V | \mathbf{d}_G(i, j) < r\}$.

Note that interesting values of C, ρ are integral between $\{0, 1, \ldots, n\}$, and it is easy to compute in $O(mn)$ time. Therefore we will assume knowledge of C, ρ 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 popular object of study Gupta et al. (2003). It can be checked that a graph with doubling dimension ρ is also a graph with polynomial growth rate ρ. 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 certain finite distance, has finite polynomial growth rate Gummadi et al. (2009).

Pair-wise graphical model and MAP. For a pair-wise Markov Random Filed (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 Σ; 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

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

where $\phi_i : \Sigma \to \mathbb{R}_+$ and $\psi_{ij} : \Sigma^2 \to \mathbb{R}_+$ are called node and edge potential functions[2]. 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

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

[2]For simplicity of the analysis we assume strict positivity of ϕ_i's and ψ_{ij}'s.

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

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

Social data and clustering/community detection. Alternatively, in a social setting, vertices of graph G can represents 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 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, to select a disjoint partition of V given G, it is not clear what is the appropriate criteria. Newman Newman (2006) proposed the notion of modularity as a criteria. The intuition behind it is that a cluster or community should be as distinct as possible from being 'random'. 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

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

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, $V = S_1 \cup \cdots \cup S_\ell$ for some $1 \leq \ell \leq n$ with $S_i \cap S_j = \emptyset$ for $i \neq j$, is given by

$$\mathcal{M}(S_1, \ldots, S_\ell) = \frac{1}{2m} \Big(\sum_{i=1}^{\ell} M(S_i) \Big). \tag{3}$$

The modularity optimization approach Newman (2006) 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 $\chi : V \to \{1, \ldots, n\}$, 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 χ and vice versa. Therefore, modularity optimization is equivalent to finding a coloring χ such that its modularity $\mathcal{M}(\chi)$ is maximized, where

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

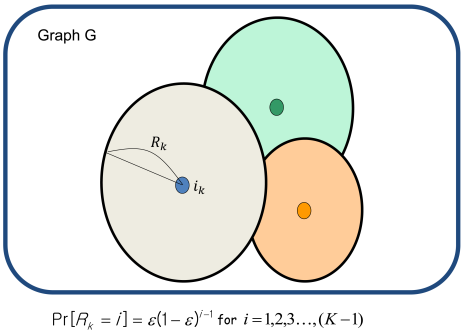


Figure 1: A pictorial description of an iteration of the graph partitioning.

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

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

3. 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 $V = V_1 \cup \cdots \cup V_p$ so that each partition $V_j, 1 \leq j \leq p$ is small. We will specify the values of K and ε in Section 4. The meta-algorithm uses an existing centralized algorithm to 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 result assignment leads to a candidate solution for the problem on entire graph. As we establish in Section 4, this becomes a pretty good solution. Next, we describe the algorithm in detail.

Step 1. Partition. We wish to create a partition of $V = V_1 \cup \cdots \cup V_p$ 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 $i_1, \ldots, i_n$. 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

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

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 $V_1 \cup \cdots \cup V_{k-1}$. 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 $V = V_1 \cup \cdots \cup V_p$ for some $p \leq n$. A caricature of an iteration is described in Figure 1.

Step 2. Merge (solving the problem). Given the partition $V = V_1 \cup \cdots \cup V_p$, 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 $G_1, \ldots, G_k$ separately. Specifically, let $\mathcal{A}$ be an algorithm for MAP or for clustering: the algorithm may be exact (e.g. one solving problem by exhaustive search over all possible options, or dynamic programming) or it may be an approximation algorithm (e.g. α-approximate for any graph). We apply $\mathcal{A}$ for each subgraph separately.

- 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 thus resulting global assignment, say $\widehat{\mathbf{x}}$ as the solution for MAP.

- 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 $G_1, \ldots, G_p$ 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 $O(Q(d^K)n)$ where $Q(\ell)$ is the computation cost of $\mathcal{A}$ for any graph with ℓ vertices.

4. Main results

4.1 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*

$$K = K(\rho, C, \delta) = \frac{8\rho}{\varepsilon} \log\left(\frac{8\rho}{\varepsilon}\right) + \frac{4}{\varepsilon} \log C + \frac{4}{\varepsilon} \log \frac{1}{\varepsilon} + 2,$$
$$\varepsilon = \varepsilon(\rho, C, \delta) = \begin{cases} \frac{\delta}{2C2^{\rho}}, & \textit{for MAP} \\ \frac{\delta}{4(2C-1)}, & \textit{for modularity optimization.} \end{cases} \tag{6}$$

Then, the following holds for the meta algorithm described in Section 3.

(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

$$(1-\delta)\mathcal{H}(\mathbf{x}^*) \leq \mathbb{E}[\mathcal{H}(\widehat{\mathbf{x}})] \leq \mathcal{H}(\mathbf{x}^*)$$
$$(1-\delta)\mathcal{M}(\chi^*) \leq \mathbb{E}[\mathcal{M}(\widehat{\chi})] \leq \mathcal{M}(\chi^*). \tag{7}$$

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

$$\left(\frac{1}{\alpha(\tilde{K})} - \delta\right) \mathcal{H}(\mathbf{x}^*) \leq \mathbb{E}[\mathcal{H}(\widehat{\mathbf{x}})] \leq \mathcal{H}(\mathbf{x}^*),$$
$$\frac{(1-\delta)}{\alpha(\tilde{K})} \mathcal{M}(\chi^*) \leq \mathbb{E}[\mathcal{M}(\widehat{\chi})] \leq \mathcal{M}(\chi^*), \tag{8}$$

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

4.2 General graph

The theorem in the previous section was for graphs with polynomial growth. We now state results for 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 $\mathcal{B} = E \setminus \cup_{k=1}^{p} E_k$),

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

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

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

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 $\psi_{ij}^L \triangleq \min_{\sigma,\sigma'\in\Sigma} \ln \psi_{ij}(\sigma, \sigma')$.

4.3 Discussion of results

Here we dissect 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.

The Theorem 1(b) suggests that, if instead of using exact procedure for each partition, when an approximation algorithm is used, 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 graph, 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, ε utilized by our partitioning procedure. For graph with polynomial growth, we provide recommendations on what the values should be for these parameters. However, for general graph, one may try various values of $K \in \{1, \ldots, n\}$ 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 $k \in \{0, \ldots, \log n\}$ and ε chosen at regular interval with granularity that an implementor is comfortable with (the smaller the granularity, the better).

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

5. Experimental Setup

In Sections 5 and 6, 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, PM was applied both on real-world networks and on synthetic networks in order 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 what value of partition radius can be determined as an appropriate value for our algorithm. All experiments were carried out using a single core[3] of an Intel i7 processor with 16GB of RAM, and PM was implemented in C++.

5.1 Datasets

5.1.1 Real Networks

To cover various aspects of the real-world networks, the experiments were conducted on various kinds of real-world networks, such as social networks (Facebook, YouTube, Live Journal), citation or collaboration networks (ArXiv, DBLP), and communication networks (Email-Enron). All the

[3] 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.

	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

Table 1: Statistics for Real-World Networks

real-world network datasets were downloaded from SNAP[4], a web site that offers the information and statistics of the networks. The network properties are summarized in Table 1. For a directed real world network, we used the corresponding undirected network in our experiments.

5.1.2 Synthetic Graphs

We also made use of synthetically generated graphs such as Grid graphs, Random k-Regular graph, Watts-Strogatz networks, and Barabási-Albert networks.

Grid Graphs. We tested our algorithm with two types of grid graph. 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.

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.

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.

Barabási-Albert Networks. We begin with the Erdös-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.

5.2 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:

[4]SNAP(http://snap.stanford.edu)

5.2.1 MAP inference

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 Kolmogorov (2006).

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

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

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 θ_i independently as per the distribution $\mathcal{U}[-1, 1]$.

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

5.2.2 Modularity Optimization

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 Girvan and Newman (2002).

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 Clauset et al. (2004).

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 Blondel et al. (2008).

6. 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 6.2. We lay out the experimental results in several changes of the network factors to determine the conditions under which our algorithm perform 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 the modularity optimization, the average modularity is simply calculated for the 100 trials for each case.

This section is organized as follows: Section 6.1 presents the overall result of the experiment. Section 6.2 describes how to obtain the proper partition radius. Sections 6.3 and 6.4 present the analyses of the experimental results on real-world networks and synthetic graphs, respectively. Section 6.5 describes the results and their explanation in regard to the two problems.

Plot. To be conducive to performance comparison, we used 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 the PM algorithm to that of the centralized algorithm on the y-axis.

6.1 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 to say, 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 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.

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.

Running Time. Running time is also prone to increase along with the partition radius. As shown in Section 3, the overall computation cost increases as an exponential function of radius. However, some experimental results show that this is not the only case; an exceptional case can be observed when large numbers of partitions are generated because of a small partition radius. Thus, we assume that it is due to the wall-clock time. 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 small 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 large growth in running time of the centralized algorithms.

6.2 Partition Radius

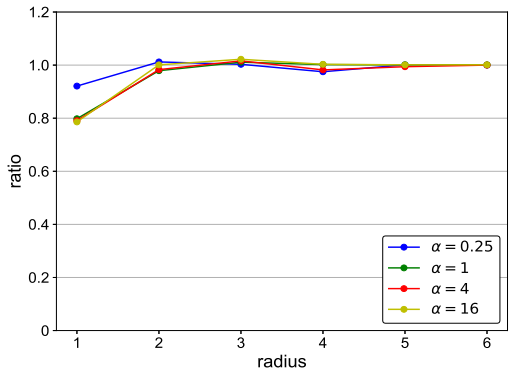
The primary question of interest: *"what value of partition radius should we choose?"* Theoretical results presented earlier in this work provide guidelines. It is definitely worth following. In addition, practically, we find a simple heuristic rule works well for "regular" enough graphs. To that end, 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 average over them. Then the following is a heuristic we suggest to choose partition radius:

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

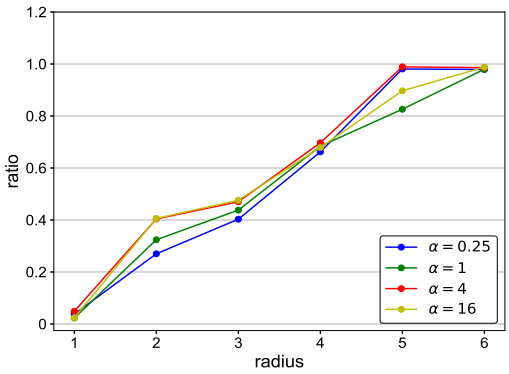
The intuition behind this rule is as follows. If the graph is "regular" enough, then the pairwise distance distribution between nodes is "uni-modal" and in which case the average distance is precisely that covers good fraction of interactions. It may make sense to choose a little smaller radius (by 1) if the graph has few very high degree nodes.

6.3 On Real-World Networks

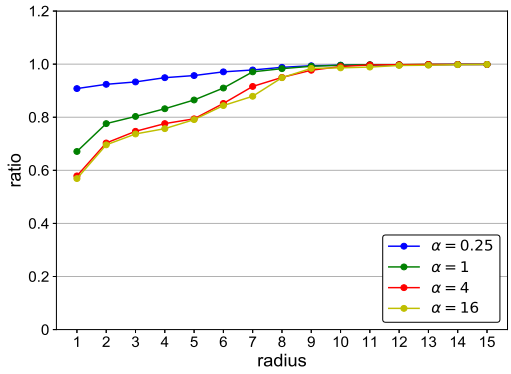
Figures 2 and 3 show the experimental results from applying our algorithm on the real-world networks. In Figure 2(a) and 2(b), on Facebook, PM finds solutions with higher energy than those obtained by using the original TRW-S within half the time in regards to MAP inference. For modularity optimization, on the same network, PM also achieves similar modularity to the original CNM in a relatively short time, as shown in Figure 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 this case R(Avg.D) - 1, which is smaller than the value of general cases, 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. Figure 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 the real-world networks.



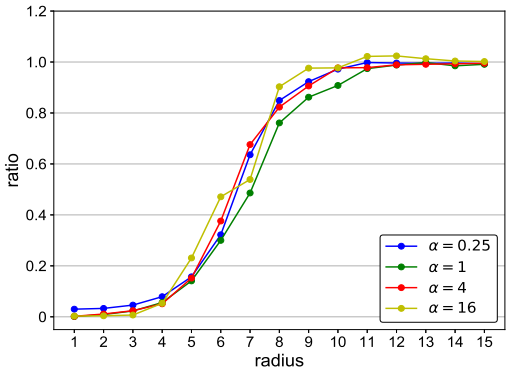
(a) Facebook *(Energy)*



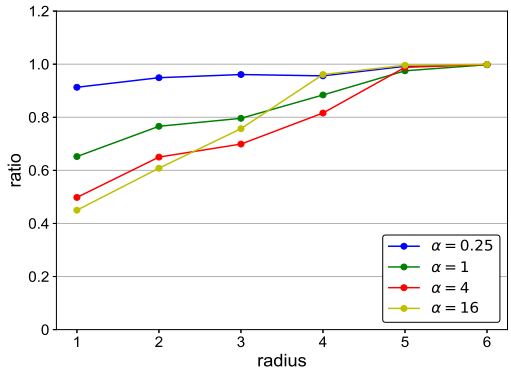
(b) Facebook *(Time)*



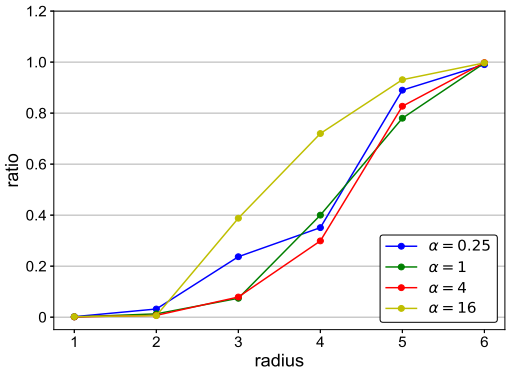
(c) ArXiv *(Energy)*



(d) ArXiv *(Time)*

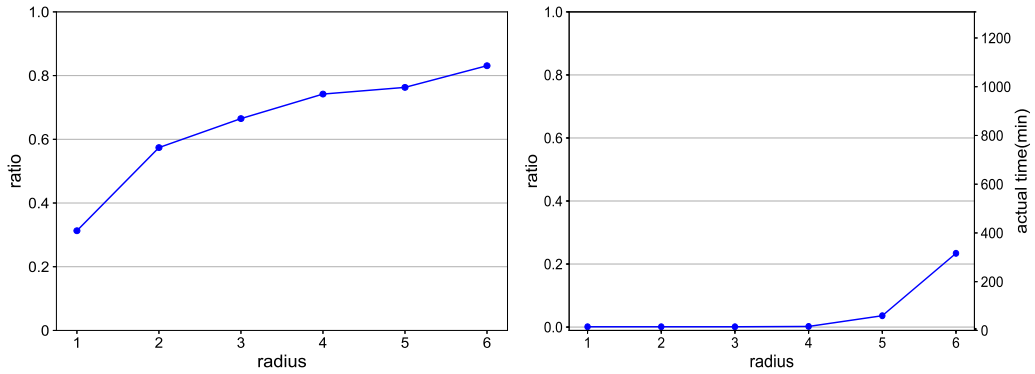


(e) Email-Enron *(Energy)*

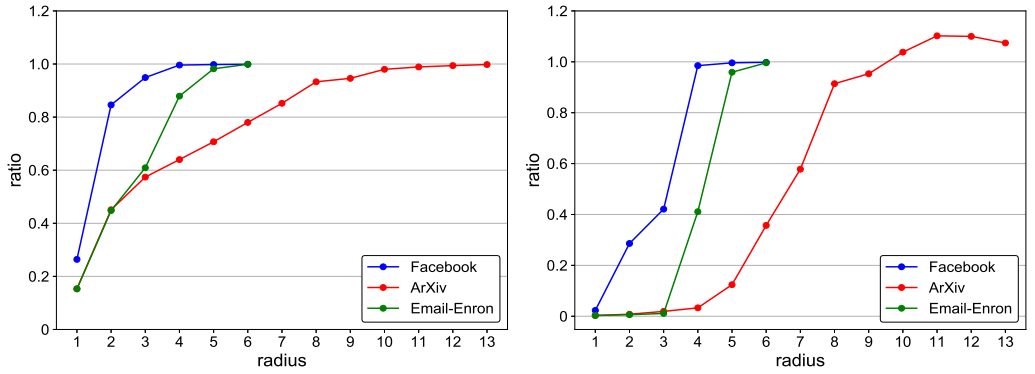


(f) Email-Enron *(Time)*

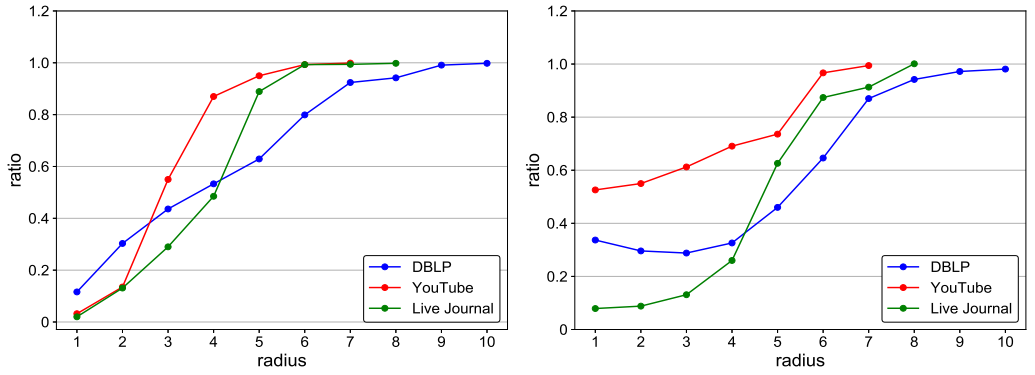
Figure 2: Real-World Networks for MAP inference



(a) ArXiv *(Energy, GN)* (b) ArXiv *(Time, GN)*



(c) Facebook, ArXiv, Email-Enron *(Energy, CNM)* (d) Facebook, ArXiv, Email-Enron *(Time, CNM)*

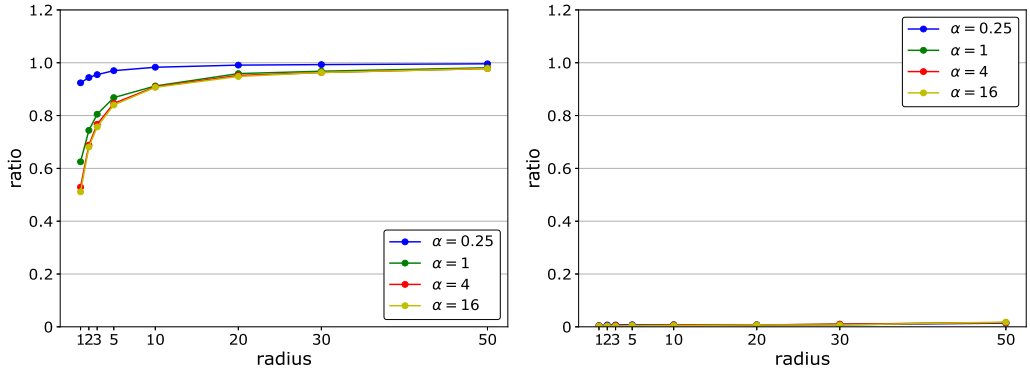


(e) ArXiv *(Energy, GN)* (f) ArXiv *(Time, GN)*

Figure 3: 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.

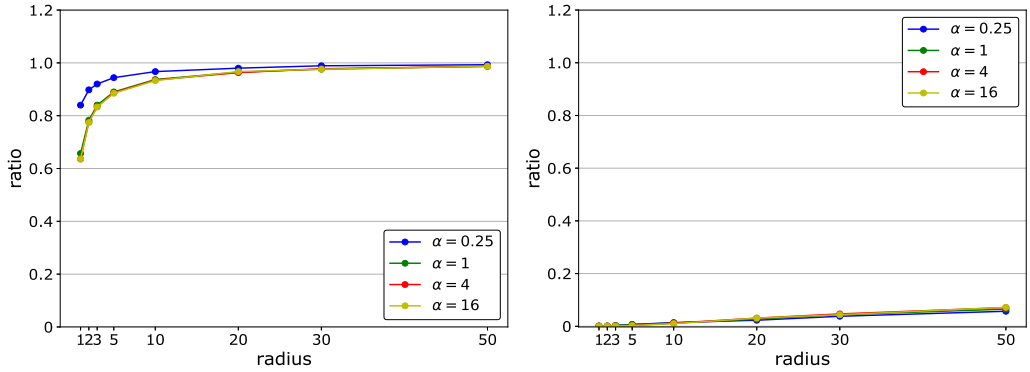
6.4 On Synthetic Graphs

On Grid Graph. PM shows a tremendous performance on the types of grid graphs stated in Section 5.1. For MAP inference, Figure 4 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 5. 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 Figure 6 and 7, PM also yields good results for modularity optimization. Moreover, PM achieves a better performance on larger graphs.



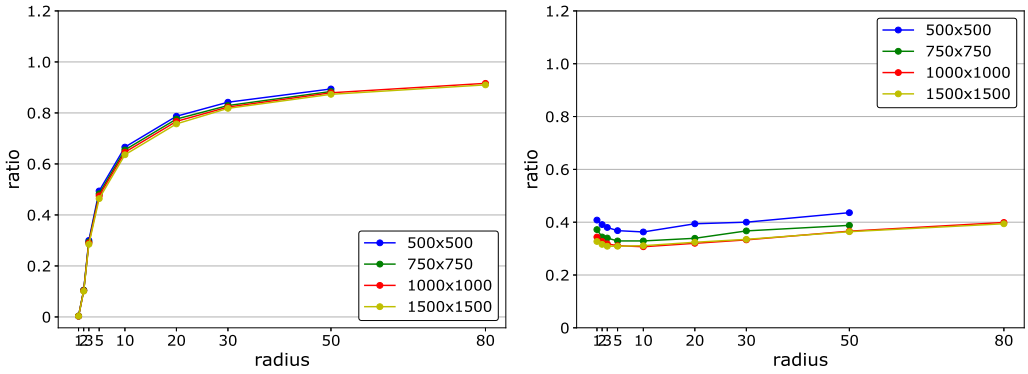
(a) Energy (b) Time

Figure 4: 1000-by-1000 Grid Graph for MAP inference



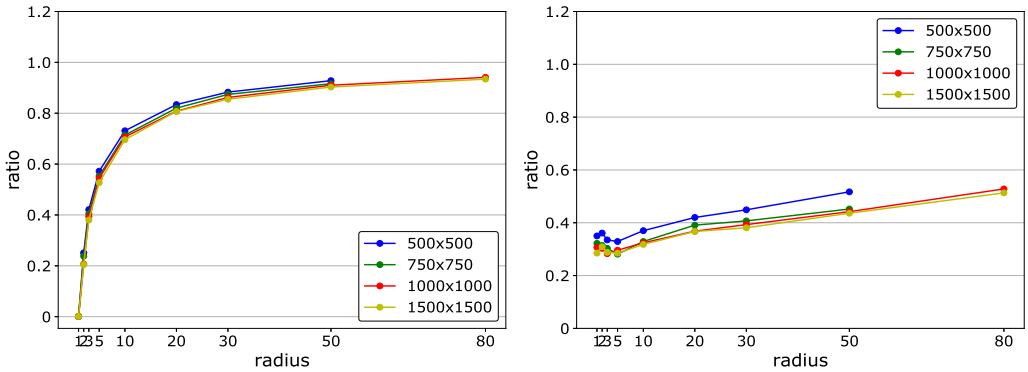
(a) Energy (b) Time

Figure 5: 1000-by-1000 Grid Graph with Diagonal edges for MAP inference



(a) Energy (b) Time

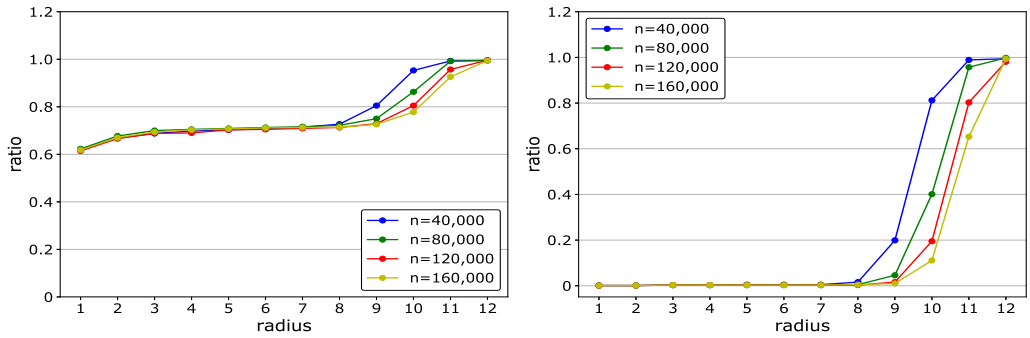
Figure 6: Grid Graphs for Modularity Optimization (*LM*)



(a) Energy (b) Time

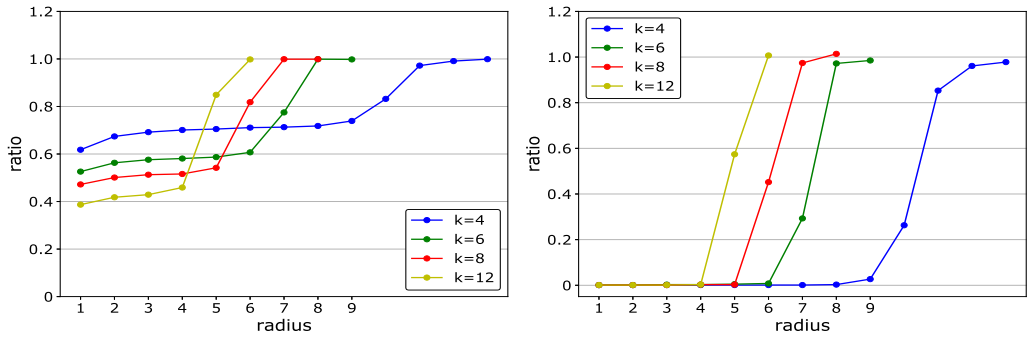
Figure 7: Grid Graphs with Diagonal edges for Modularity Optimization (*LM*)

Random k-Regular Network. For MAP inference, PM shows almost the same efficiency for a fixed degree, irrespective of the network size. In Figure 8, 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 10. For the same size of networks, the increase in degree makes our algorithm less effective, as shown in Figures 9 and 11. These results show 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.



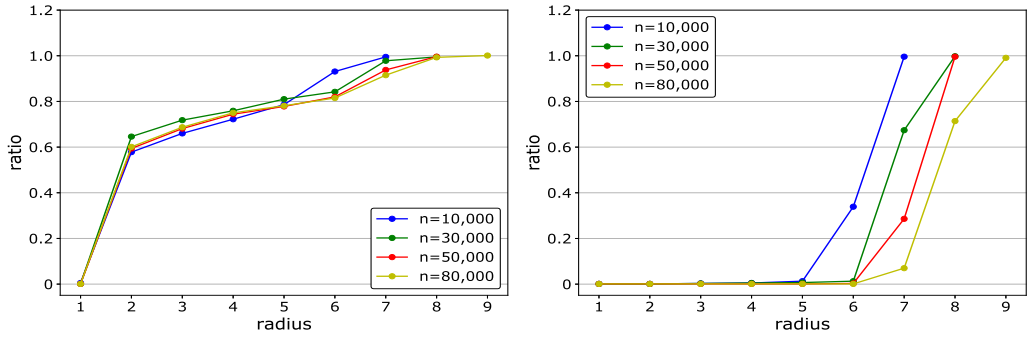
(a) Energy (b) Time

Figure 8: Random k-Regular Networks for MAP inference (For fixed $k = 4$, $\alpha = 1$)



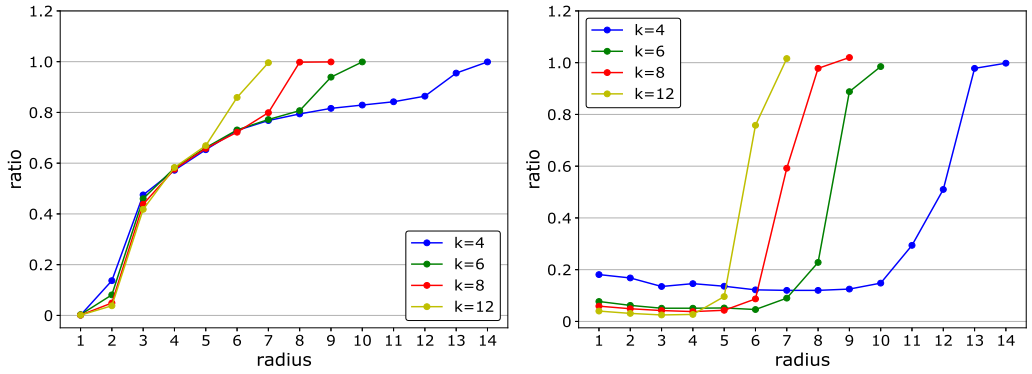
(a) Energy (b) Time

Figure 9: Random k-Regular Networks for MAP inference (For fixed $n = 10^5$, $\alpha = 1$)



(a) Energy (b) Time

Figure 10: Random k-Regular Networks for Modularity Optimization (For fixed $k = 6$ / *CNM*)

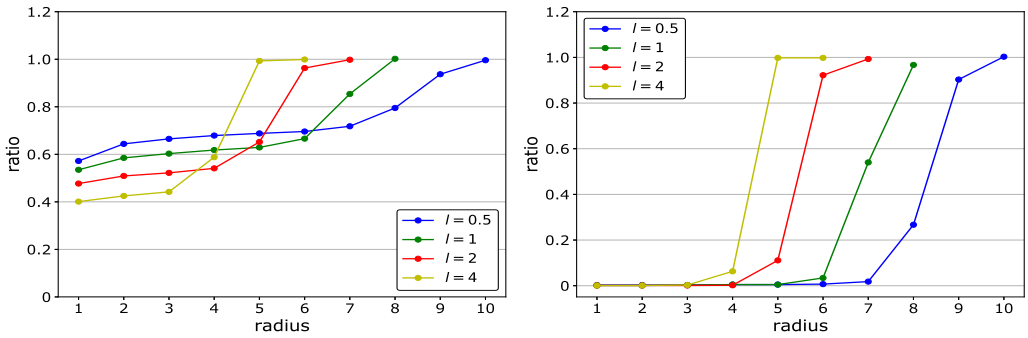


(a) Energy (b) Time

Figure 11: Random k-Regular Networks for Modularity Optimization (For fixed $n = 10^6$ / *LM*)

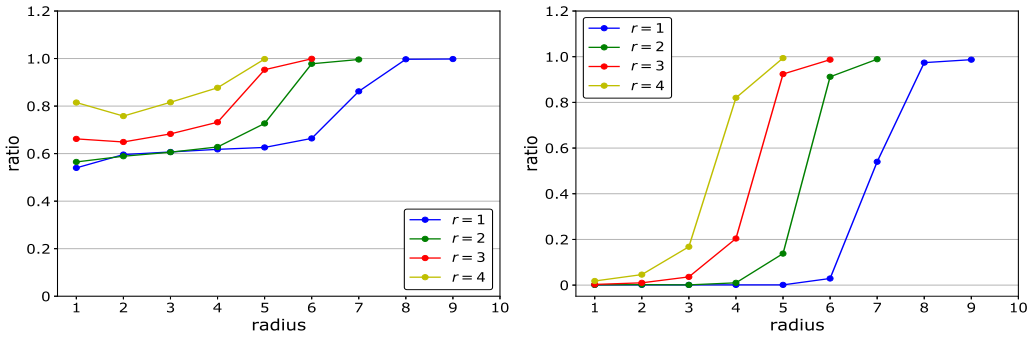
On Watts-Strogatz Network. Note that a Watts-Strogatz network has an underlying regular structure with a small amount of randomness. First, the randomness increases with the number of long distance edges. Increasing randomness could have adverse effects on our algorithm. For both the problems, the experimental results are better for our algorithm with the small number of long distance edges, as shown in Figure 12 and 14. 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 to 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 13, the results show that our algorithm is very efficient, producing nearly 81% energy compared to TRW-S but 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 largely improves the efficiency of PM, as shown in Figure 15. 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.



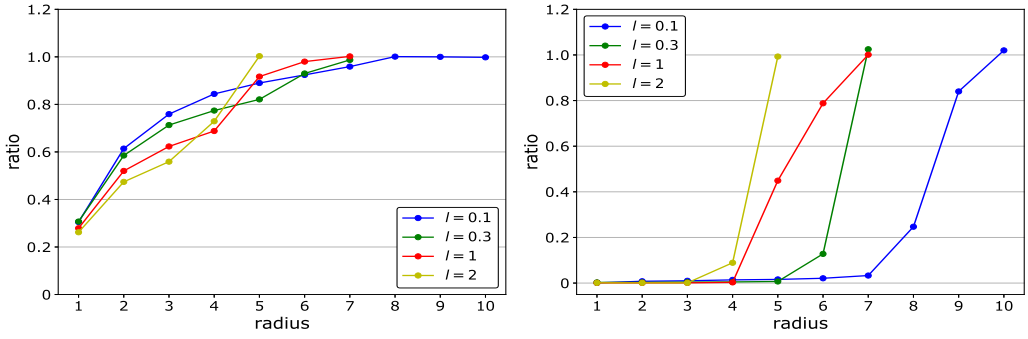
(a) Energy (b) Time

Figure 12: Watts-Strogatz Networks for MAP inference
(For fixed n = 300-by-300, $r = 1$, $\alpha = 1$)



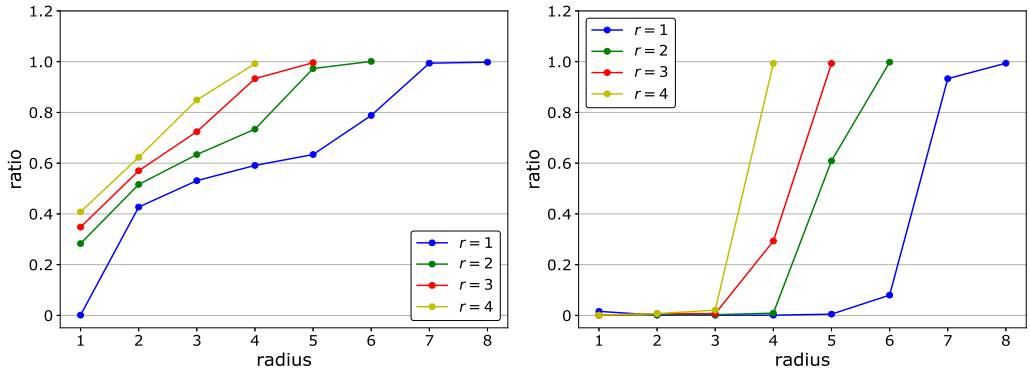
(a) Energy (b) Time

Figure 13: Watts-Strogatz Networks for MAP inference
(For fixed n = 250-by-250, $l = 1$, $\alpha = 1$)



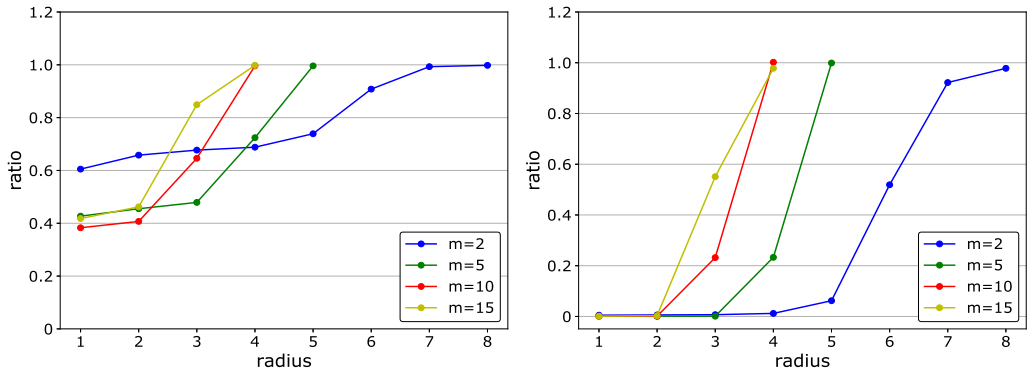
(a) Energy (b) Time

Figure 14: Watts-Strogatz Networks for Modularity Optimization
(For fixed n = 150-by-150, $r = 2$ / *CNM*)



(a) Energy (b) Time

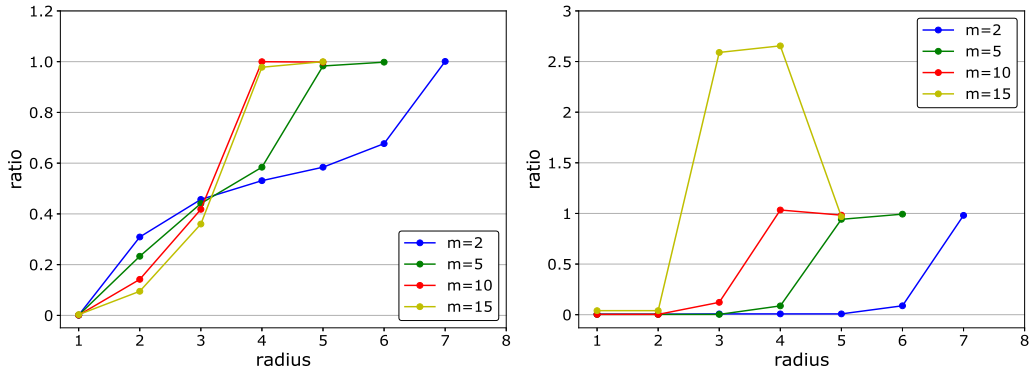
Figure 15: Watts-Strogatz Networks for Modularity Optimization
(For fixed n = 100-by-100, l = 1 / *CNM*)



(a) Energy (b) Time

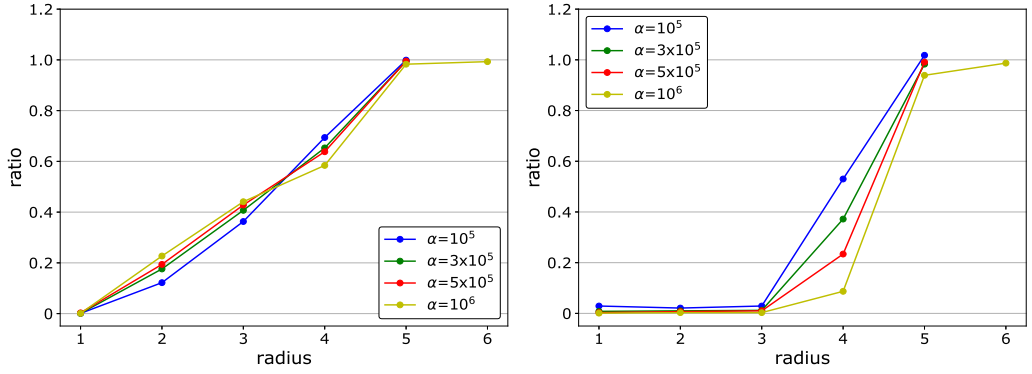
Figure 16: Barabási-Albert Networks for MAP inference (For fixed n = 50,000, $\alpha = 1$)

On Barabási-Albert Network. For both the problems, as the number of additional edges at each time step increases, the advantage of our algorithm decreases. The results for this case are shown in Figures 16 and 16. Hubs could exist due to the generative processes of Barabási-Albert network. These hubs greatly 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 18, the results show that increasing network size brings about distributed effects that are more efficient.



(a) Energy (b) Time

Figure 17: Barabási-Albert Networks for Modularity Optimization (For fixed $n = 10^6$ / *LM*)



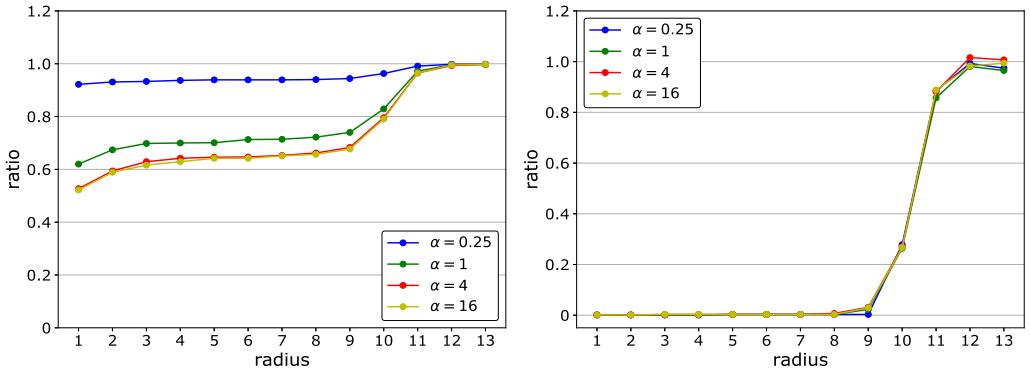
(a) Energy (b) Time

Figure 18: Barabási-Albert Networks for Modularity Optimization (For fixed $m = 5$ / *LM*)

6.5 Discussion for each problem

6.5.1 MAP inference

Interaction parameter α. We defined energy functions of discrete variables with unary and pairwise terms, where parameter α of (11) determines the strength of relationship between pairs of nodes. Accordingly, the increasing value of α assigns a larger weight to edges, which consequently incur more errors caused by our partitioning scheme. Therefore, our algorithm generally yields better approximation when α has a relatively small value, as shown in Figure 19. However, it requires comparatively more time for better assignment when α has a small value. This is because the amount of reduced errors decreases as the partitions grow.



(a) Energy (b) Time

Figure 19: Results as the value of α changes
Random k-Regular Networks for MAP (For fixed $n = 10^5$, $k = 4$)

Average Degree. It is observed that MAP inference is more affected by the average degree than in the case of modularity optimization. When partitions are very small, there is a tendency to produce better approximation as the average degree gets bigger. In particular, when the average degree is very high, our algorithm obtains outstanding results even for very small partitions, such as in Facebook. However, one cannot always obtain good results with a high average degree. As appears by our general experimental results, the increase of average degree that leads to an increase in randomness can negatively influence our algorithm. Our algorithm accomplishes better efficiency generally when the increase of average degree improves the regularity. However, in like manner to α, 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 α has a small value with a high average degree. However, our algorithm can only realize positive effects when the increase of average degree improves the regularity. In other words, PM shows better results when applied on well-distributed regular networks, while taking less consideration of the relationship between nodes.

6.5.2 Modularity optimization

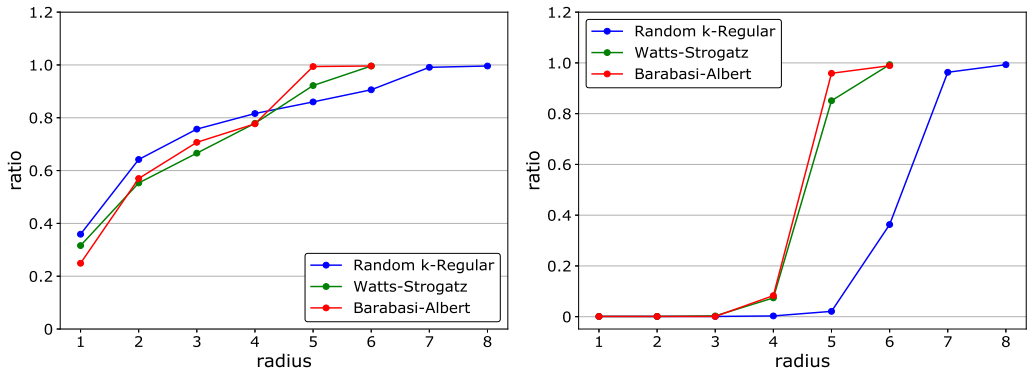
Due to its high computational complexity, it is difficult for GN to analyze networks under several conditions. Accordingly, we present the experimental results of GN separately, and PM shows a good performance in this case, as shown in Figure 20. In addition, we observe that the three centralized algorithms we used to extract the community structure show somewhat different tendencies when it comes to the change in the partition radius. Focusing on these differences, we split the analysis of the experimental results into two parts for closer examination:

Modularity. As shown in Figure 21(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 reduces the error most rapidly as the partition radius increases.

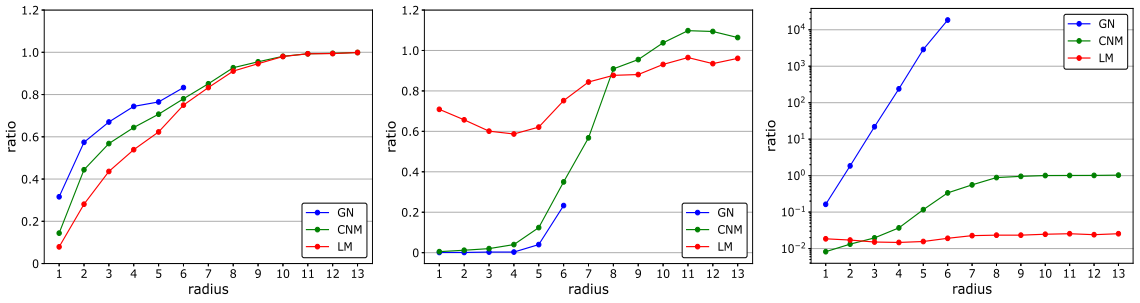
Running Time. In Figure 21(b) and 21(c), GN and CNM both show a tendency of dramatically increasing running time gap between two consecutive radius steps along with partition radius. Compared to GN and CNM, LM shows a relatively small change of the gap. When the algorithm with high computational complexity 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 to say, 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 that they can adopt. PM enables them to analyze the networks that had been 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 difficult to tackle large partitions.



(a) Energy (b) Time

Figure 20: The Experimental Results for Girvan and Newman algorithm



(a) Energy (b) Time (c) Time

Figure 21: A Comparison on Results of three algorithms for Modularity Optimization

7. 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. The resulting distributed implementation, with the underlying graph having polynomial growth property, 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.

Acknowledgments

Part of this work appeared in the preliminary version Jung et al. (2009). This work is supported in parts by Army Research Office under MURI Award 58153-MA-MUR, ARO MURI 133668-5079809, NSF CMMI-1462158 and in part by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2012032786).

Appendix A. Proofs of Theorems 1, 2

A.1 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 will be used in the proofs of Theorems 1, 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 $e \in E$,*

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

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*

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

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 Σ arbitrarily. Note that for above assignment to be possible, we have used matching property of M. Therefore, we have

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

Here (a) follows because ψ_{ij}, ϕ_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. □

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 $\widehat{\mathbf{x}}$ 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 the partition scheme produces output $\widehat{\mathbf{x}}$ such that*

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

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 $S_1, \ldots, S_K$ 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 $\widehat{\mathbf{x}}$ be the output of the partition scheme, and let $\widehat{\mathbf{x}}_k$ be the restriction of $\widehat{\mathbf{x}}$ to the component S_k. Note that since $\widehat{\mathbf{x}}_k$ is a MAP assignment of $\mathcal{H}_k(\cdot)$ by the definition of our algorithm, for all $k = 1, 2, \ldots K$,

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

Now, we have

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

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

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

$$\begin{aligned}
\mathbb{E}[\mathcal{H}(\widehat{\mathbf{x}})] &\overset{(a)}{\geq} \mathcal{H}(\mathbf{x}^*) - \mathbb{E}\left[\sum_{(i,j)\in\mathcal{B}} \left(\psi_{ij}^U - \psi_{ij}^L\right)\right] \qquad (17) \\
&\overset{(b)}{\geq} \mathcal{H}(\mathbf{x}^*) - 2\varepsilon\left(\sum_{(i,j)\in E} \left(\psi_{ij}^U - \psi_{ij}^L\right)\right) \qquad (18) \\
&\overset{(c)}{\geq} \mathcal{H}(\mathbf{x}^*)\left(1 - 2\varepsilon(d^* + 1)\right) \qquad (19) \\
&\overset{(d)}{\geq} (1-\delta)\mathcal{H}(\mathbf{x}^*). \qquad (20)
\end{aligned}$$

Here (a) follows from Lemma 3, (b) follows from Lemma 1, (c) from Lemma 2, and (d) follows from the definition of ε 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.

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

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

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

Hence we have that

$$\begin{aligned}
\mathbb{E}[\mathcal{H}(\widehat{\mathbf{x}})] &\geq \frac{1}{\alpha(\tilde{K})}\mathcal{H}(\mathbf{x}^*) - \mathbb{E}\left[\sum_{(i,j)\in\mathcal{B}}\left(\psi_{ij}^U - \psi_{ij}^L\right)\right] && (23)\\
&\overset{(a)}{\geq} \frac{1}{\alpha(\tilde{K})}\mathcal{H}(\mathbf{x}^*) - 2\varepsilon\left(\sum_{(i,j)\in E}\left(\psi_{ij}^U - \psi_{ij}^L\right)\right) && (24)\\
&\overset{(b)}{\geq} \mathcal{H}(\mathbf{x}^*)\left(\frac{1}{\alpha(\tilde{K})} - 2\varepsilon(d^*+1)\right) && (25)\\
&\overset{(c)}{\geq} \Big(\frac{1}{\alpha(\tilde{K})} - \delta\Big)\mathcal{H}(\mathbf{x}^*). && (26)
\end{aligned}$$

Here (a) follows from Lemma 1, (b) follows from Lemma 2, and (c) from the definition of ε 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 $\mathcal{B} = E\backslash \cup_{k=1}^{p} E_k$ 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 $\mathcal{G} = (V, \mathbf{d_G})$ 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 $\mathbf{d_G}(i_1, e) < K$, 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

$$E_{11} = \{\mathbf{d_G}(i_1, e) < R_1\},\ E_{12} = \{\mathbf{d_G}(i_1, e) = R_1\},\ \text{and}\ E_{13} = \{\mathbf{d_G}(i_1, e) > R_1\}.$$

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 $\mathbf{d_G}$ is still considered as the metric on the points[5] of $\mathcal{W}_1$. By its definition, the partition scheme excluding the first iteration is the same as the partition scheme applied to $(\mathcal{W}_1, \mathbf{d_G})$. 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 $(\mathcal{W}_1, \mathbf{d_G})$ 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 $\mathbf{d_G}(i_1, e) = \ell < K$. By the definition of probability distribution of $\mathbf{Q}$, we have

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

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

That is,

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

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

$$\begin{aligned} \mathbb{P}[e \in \mathcal{B}|E_1] &= \mathbb{P}[e \in \mathcal{B}|E_{11} \cap E_1]\mathbb{P}[E_{11}|E_1] + \mathbb{P}[e \in \mathcal{B}|E_{12} \cap E_1]\mathbb{P}[E_{12}|E_1] \\ &\quad + \mathbb{P}[e \in \mathcal{B}|E_{13} \cap E_1]\mathbb{P}[E_{13}|E_1] \\ &\leq 0 \times q + 1 \times \frac{\varepsilon q}{1-\varepsilon} + (\varepsilon + P_K|\mathbf{B}(e, K)|)\left(1 - \frac{q}{1-\varepsilon}\right) \\ &= \varepsilon + P_K|\mathbf{B}(e, K)| + \frac{q}{1-\varepsilon}\left(\varepsilon - \varepsilon - P_K|\mathbf{B}(e, K)|\right) \\ &= \varepsilon + P_K|\mathbf{B}(e, K)| - \frac{qP_K|\mathbf{B}(e, K)|}{1-\varepsilon} \\ &\leq \varepsilon + P_K|\mathbf{B}(e, K)|. \end{aligned} \tag{29}$$

[5]Note the following subtle but crucial point. We are not changing the metric $\mathbf{d_G}$ after we remove points from the original set of points.

Case 2. Now, suppose $i_1 \in V$ is such that $\mathbf{d_G}(i_1, e) = K$. 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,

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

Under the event $E_{21}^c \cap E_2$, we have $e \in \mathcal{W}_1$, and the remaining metric space $(\mathcal{W}_1, \mathbf{d_G})$. 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

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

From (30) and (31), we have

$$\begin{aligned}\mathbb{P}[e \in \mathcal{B}|E_3] &\leq P_K + (1 - P_K)(\varepsilon + P_K \cdot (|\mathbf{B}(e, K)| - 1)) \\ &= \varepsilon(1 - P_K) + P_K|\mathbf{B}(e, K)| + P_K^2(1 - |\mathbf{B}(e, K)|) \\ &\leq \varepsilon + P_K|\mathbf{B}(e, K)|. \end{aligned}$$

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 $\mathbf{d_G}(i_1, e) > K$. Then, at the end of the first iteration of the algorithm, we again have the remaining metric space $(\mathcal{W}_1, \mathbf{d_G})$ 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.

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

This completes the proof of Proposition 1. □

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.

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

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

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

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:

$$\frac{K}{2} \geq \frac{2}{\varepsilon}\log C + \frac{2}{\varepsilon}\log\frac{1}{\varepsilon} + 1. \tag{35}$$

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

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

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

$$\frac{K}{2} \geq \frac{2\rho}{\varepsilon}\log K. \tag{36}$$

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

A.2 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,*

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

Proof. Since the graph has polynomial growth with degree ρ 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 $\frac{1}{2m}\big(1 - \frac{C^2}{2m}\big)$ 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 $\frac{1}{2(2C-1)}\big(1 - \frac{C^2}{2m}\big)$. This completes the proof of Lemma 5 (Similar result, with tighter constant, follows from Han (2008)). □

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 $V = V_1 \cup \cdots \cup V_p$ 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 χ^k be a coloring (clustering) of G_k with maximum modularity. Let χ^* be a coloring of G with maximum modularity ($\mathcal{M}^*$) and let $\chi^{*,k}$ be the restriction of χ^* to G_k. Let $\widehat{\chi}$ denote the clustering of G obtained by taking union of clusterings $\chi^1, \ldots, \chi^p$. Then we claim the following.

Lemma 6. *For any partition $V = V_1 \cup \cdots \cup V_p$,*

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

Proof. Consider the following:

$$\begin{aligned}
2m\,\mathcal{M}(\widehat{\chi}) &= \sum_{i,j\in V} \mathbf{1}_{\{\widehat{\chi}(i)=\widehat{\chi}(j)\}}\Big(A_{ij} - \frac{d_i d_j}{2m}\Big) \overset{(a)}{=} \sum_{k=1}^p \sum_{i,j\in V_k} \mathbf{1}_{\{\chi^k(i)=\chi^k(j)\}}\Big(A_{ij} - \frac{d_i d_j}{2m}\Big) \\
&\overset{(b)}{\geq} \sum_{k=1}^p \sum_{i,j\in V_k} \mathbf{1}_{\{\chi^{*,k}(i)=\chi^{*,k}(j)\}}\Big(A_{ij} - \frac{d_i d_j}{2m}\Big) \\
&= \sum_{i,j\in V} \mathbf{1}_{\{\chi^*(i)=\chi^*(j)\}}\Big(A_{ij} - \frac{d_i d_j}{2m}\Big) - \sum_{(i,j)\in V^2 \setminus \cup_{k=1}^p V_k^2} \mathbf{1}_{\{\chi^*(i)=\chi^*(j)\}}\Big(A_{ij} - \frac{d_i d_j}{2m}\Big) \qquad (37) \\
&\geq \sum_{i,j\in V} \mathbf{1}_{\{\chi^*(i)=\chi^*(j)\}}\Big(A_{ij} - \frac{d_i d_j}{2m}\Big) - |E \setminus \cup_{k=1}^p E_k|, \qquad (38)
\end{aligned}$$

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 $E \backslash \cup_{k=1}^{p} E_k$. 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 χ^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 χ^* to G_k. This completes the proof of Lemma 6 since the first term in (38) is precisely $2m\mathcal{M}(\chi^*) = 2m\mathcal{M}^*$. □

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

$$\mathcal{M}(\widehat{\chi}) \geq \mathcal{M}(\chi^*)\Big(1 - \frac{\beta}{2\mathcal{M}(\chi^*)}\Big) \geq \mathcal{M}(\chi^*)\Big(1 - 2(2C-1)\beta\Big). \tag{39}$$

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

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

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 $\tilde{\chi}^1, \ldots, \tilde{\chi}^p$ be the clustering (coloring) produced by $\mathcal{A}$ on graphs $G_1, \ldots, G_p$. 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 $\tilde{\chi}^1, \ldots, \tilde{\chi}^p$, we have

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

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).

References

B. Awerbuch, M. Luby, A.V. Goldberg, and S.A. Plotkin. Network decomposition and locality in distributed computation. In *Foundations of Computer Science (FOCS)*. IEEE, 1989.

M. Bayati, D. Shah, and M. Sharma. Maximum weight matching via max-product belief propagation. In *IEEE ISIT*, 2005.

M. Bayati, D. Shah, and M. Sharma. Max-Product for Maximum Weight Matching: Convergence, Correctness, and LP Duality. *IEEE Transactions on Information Theory*, 54(3):1241–1251, 2008.

V. Blondel, G. Krings, and I. Thomas. Regions and borders of mobile telephony in belgium and in the brussels metropolitan zone. *Brussels Studies*, 42(4), 2010.

V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008, 2008.

A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(066111), 2004.

B. DasGupta and D. Desai. On the complexity of newman's community finding approach for biological and social networks. *Arxiv preprint arXiv:1102.0969*, 2011.

M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

R. Gummadi, K. Jung, D. Shah, and R. Sreenivas. Computing the capacity region of a wireless network. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

A. Gupta, R. Krauthgamer, and J.R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Foundations of Computer Science (FOCS)*, 2003.

Y. Han. Matching for graphs of bounded degree. *Frontiers in Algorithmics*, pages 171–173, 2008.

A. Hassidim, J.A. Kelner, H.N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Foundations of Computer Science (FOCS)*, 2009.

B. Huang and T. Jebara. Loopy belief propagation for bipartite maximum weight b-matching. *Artificial Intelligence and Statistics (AISTATS)*, 2007.

K. Jung and D. Shah. Local algorithms for approximate inference in minor-excluded graphs. In *Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.

K. Jung, P. Kohli, and D. Shah. Local rules for global map: When do they work? *Advances in Neural Information Processing Systems (NIPS)*, 22:871–879, 2009.

P. Klein, S.A. Plotkin, and S. Rao. Excluded minors, network decomposition, and multicommodity flow. In *ACM symposium on Theory of computing (STOC)*, 1993.

Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1568–1583, 2006.

M.E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577, 2006.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann, 1988.

D. Peleg. *Distributed computing: a locality-sensitive approach*, volume 5. Society for Industrial Mathematics, 2000.

S. Sanghavi, D. Shah, and A. Willsky. Message-passing for Maximum Weight Independent Set. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

D. Sontag and T. Jaakkola. Tree block coordinate descent for map in graphical models. *Journal of Machine Learning Research - Proceedings Track*, 5:544–551, 2009.

R. Swendsen and J. Wang. Nonuniversal critical dynamics in monte carlo simulations. *Phys. Rev. Letter.*, 58:86–88, 1987.

D.l Tarlow, D. Batra, P. Kohli, and V. Kolmogorov. Dynamic tree block coordinate ascent. In *ICML*, 2011.

M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 2005.

J. Yedidia, W. Freeman, and Y. Weiss. Generalized belief propagation. *Mitsubishi Elect. Res. Lab., TR-2000-26*, 2000.