Towards Optimal Community Detection: From Trees to General Weighted Networks

Thang N. Dinh and My T. Thai
Computer & Information Science & Engineering
University of Florida, Gainesville, FL 32611
Email: {tdinh, mythai}@cise.ufl.edu

Abstract

Many networks including the Internet, social networks, and biological relations are found to be naturally divided into communities of densely connected nodes, known as community structure. Since Newman's suggestion of using *modularity* as a measure to qualify the goodness of community structures, many efficient methods to maximize modularity have been proposed but without *optimality guarantees*. In this paper, we study exact and theoretically near-optimal algorithms for maximizing modularity. In the first part, we investigate the complexity and approximability of the problem on tree graphs. Somewhat surprisingly, the problem is still NP-complete on trees. We then provide a polynomial time algorithm for uniform-weighted trees, a pseudo-polynomial time algorithm and a PTAS for trees with arbitrary weights. In the second part, we propose *sparse metric*, a set of linear programming formulations for general graphs. By exploiting the graph connectivity structure, sparse metrics helps to reduce substantially the number of constraints, thus, vastly improve the running time for solving linear programming and integer programming. As a result, networks of thousands of vertices can be solved in minutes while the current largest instance solved with mathematical programming has less than 250 vertices.

1 Introduction

Many complex systems of interest such as the Internet, social, and biological relations, can be represented as networks consisting a set of *nodes* which are connected by *edges* between them. Research in a number of academic fields has uncovered unexpected structural properties of complex networks including small-world phenomenon [1], power-law degree distribution [2], and the existence of community structure [3] where nodes are naturally clustered into tightly connected modules, also known as communities, with only sparser connections between them. Finding this community structure is a fundamental but challenging problem in the study of network systems and not yet satisfactorily solved, de-

spite the huge effort of a large interdisciplinary community of scientists working on it over the past few years [4].

The ability to detect such communities can be of significant practical importance, providing insight into how network function and topology affect each other. For instance, communities within the World Wide Web may correspond to sets of web pages on related topics; communities within mobile networks may correspond to sets of friends or colleagues; in computer networks communities may correspond to users in a peer-to-peer traffic, or botnet farms [5]. Detecting this special sub-structure also finds itself extremely useful in deriving social-based applications such as forwarding and routing strategies in communication networks [6–8], sybil defense [9, 10], worm containment on cellular networks [7, 11], and sensor programming [12].

Newman-Girvan's modularity that measures the "strength" of partition of a network into modules (also called communities or clusters) [13] has rapidly become an essential element of many community detection methods. Modularity is defined as the fraction of the edges that fall within the given communities minus the expected such fraction if edges were distributed at random. One can search for community structure precisely by looking for the divisions of a network that have positive, and preferably large, values of the modularity. This is the main motivation for numerous optimization methods that find communities in the network via maximizing modularity as surveyed in [4]. Unfortunately, Brandes et al. [14] have shown that modularity maximization is an NP-hard problem. Thus, it is desirable to design algorithms maximizing modularity that provide lower bounds on the modularity values.

In contrast to the vast amount of work on maximizing modularity, the only known polynomial-time approach to find a good community structure with guarantees is due to Agarwal and Kempe [15] in which they rounded the fractional solution of a linear programming (LP). The value obtained by solving the LP gives an upper bound on the maximum achievable modularity. The main drawback of the approach is the large LP formulation that consumes both time and memory resources. As shown in their paper, the approach can only be used on the networks of up to 235 nodes. In addition, no approximation ratio was proven for the proposed algorithms.

In this paper, we study exact and approximation algorithms for maximizing modularity. In the first part, we investigate the complexity and approximability of the problem on tree graphs. Somewhat surprisingly, the problem is still NP-complete on trees. We then provide a polynomial time algorithm for uniform-weighted trees, a pseudo-polynomial time algorithm and a PTAS for trees with arbitrary weights. In the second part, we address the main drawback of the rounding LP approach by introducing a new formulation, called *sparse metric*. We show both theoretically and experimentally that the new formulation substantially reduces the time and memory requirements without sacrificing on the quality of the solution. The size of solved network instances raises from hundred to several thousand nodes while the running time on the medium-instances are sped up from 10 to 150 times. In fact, the modularity values found by rounding LP are optimal for many network instances.

Related work. A vast amount of methods to find community structure is surveyed in [4]. Brandes et. al. proves the NP-completeness for maximizing modularity, the first hardness result for this problem. DasGupta et. al. recently show that maximizing modularity is APX-hard, i.e., it admits no polynomial-time approximation schemes (PTAS) [16].

Modularity has several known drawbacks. Fortunato and Barthelemy [17] has shown the resolution limit i.e. maximizing modularity methods fail to detect communities smaller than a scale, the resolution limit only appears when the network is substantially large [18]. Another drawback is modularitys highly degenerate energy landscape [19], which may lead to very different yet equally high modularity partitions. However, for small and medium networks of several thousand nodes, the method proposed by Blondel et al. [20] to optimize modularity is one of the best performing algorithms according to the LFR benchmark [18]. Thus our proposed method is applicable whenever high quality solutions for small and medium networks are desired. Approximation algorithms for maximizing modularity are first studied in [21] for scale-free networks and in [16] for d-regular networks.

Organization. We present definitions and notions in Section 2. In Section 3, we present the first part of our results on maximizing modularity over tree graphs including the NP-completeness, the polynomial-time algorithm for uniform-weighted trees, the pseudo-polynomial time algorithm and a PTAS for general weighted trees. We propose in Section 4 the second part of our results: the *sparse metric*[1] technique to efficiently maximize modularity via rounding the linear programming. exact algorithm on tree is presented in Section 3. We show experimental results for the *sparse metric* in Section 5 to illustrate the time efficiency of our formulations over the previous approach.

2 Preliminaries

We consider a network represented as an undirected graph $G = (V, E)$ consisting of $n = |V|$ vertices and $m = |E|$ edges. The *adjacency matrix* of G is denoted by $\boldsymbol{A} = (A_{ij})$, where A_{ij} is the weight of edge (i, j) and $A_{ij} = 0$ if $(i, j) \notin E$. We also denote the (weighted) degree of vertex i, the total weights of edges incident at i, by $\mathsf{deg}(i)$ or, in short, d_i.

Community structure (CS) is a division of the vertices in V into a collection of disjoint subsets of vertices $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$ that the union gives back V. Each subset $C_i \subseteq V$ is called a *community* and we wish to have more edges connecting vertices in the same communities than edges that connect vertices in different communities. The *modularity* [23] of $\mathcal{C}$ is defined as

$$Q(\mathcal{C}) = \frac{1}{2M} \sum_{i,j \in V} (A_{ij} - \frac{d_i d_j}{2M}) \delta_{ij} \tag{1}$$

[1] A weaker version of the sparse metric, which doubles the number of constraints, were introduced in [22] for the β-disruptor problem.

where d_i and d_j are degree of nodes i and j, respectively, M is the total edge weights and the element δ_{ij} of the membership matrix $\boldsymbol{\delta}$ is defined as

$$\delta_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in the same community} \\ 0, & \text{otherwise.} \end{cases}.$$

The modularity values can be either positive or negative and the higher (positive) modularity values indicate stronger community structure. The *maximizing modularity problem* asks to find a division which maximize the modularity value.

We also define the modularity matrix $\boldsymbol{B}$ [23] with entries $B_{ij} = A_{ij} - \frac{d_i d_j}{2M}$. Then, the modularity can conveniently be written in the matrix form as

$$Q(\mathcal{C}) = \frac{1}{2M}\text{trace}(B\delta)$$

Alternatively the modularity can be equivalently defined as

$$Q(\mathcal{C}) = \sum_{t=1}^{l} \left(\frac{E(C_t)}{M} - \frac{\mathsf{vol}(C_t)^2}{4M^2} \right), \tag{2}$$

where $E(C_t)$ is the total weight of edges whose endpoints are in C_t and for an $S \subseteq V$ we define $\mathsf{vol}(S) = \sum_{v \in S} d_v$, the volume of S.

3 Complexity and Approximation Algorithms on Trees

We first show that maximizing modularity is even NP-hard on tree graphs. Then we propose a polynomial-time dynamic programming algorithm for the problem when the tree has uniform weights and its extension into a pseudo-polynomial time algorithm to solve the problem on trees with arbitrary weights. The existence of the pseudo-polynomial time algorithm implies that maximizing modularity on trees is a weakly NP-complete problem. Finally we propose a PTAS for the problem that finds a CS with modularity at least $(1 - \epsilon)$ the maximum modularity within $O(n^{1+1/\epsilon})$.

3.1 NP-completeness

It has been proved in [14] that maximizing modularity is hard. We further prove that the decision version of maximizing modularity is still hard on trees, one of the simplest graph classes. Our proof is much simpler than the first proof in [14], while implying the NP-hardness on a more restricted graph class.

Definition 1 (Modularity on Trees)**.** *Given a tree $T = (V, E)$, weights $c(e) \in \mathbb{Z}^+$ for edges $e \in E$, and a number K, is there a community structure $\mathcal{C}$ of T, for which $Q(\mathcal{C}) \geq K$?*

Our hardness result is based on a transformation from the following decision problem.

Definition 2 (SUBSET-SUM)**.** *Given a set of k positive integers $w_1, w_2, \ldots, w_k$ and an integer S, does any non-empty subset sum to S?*

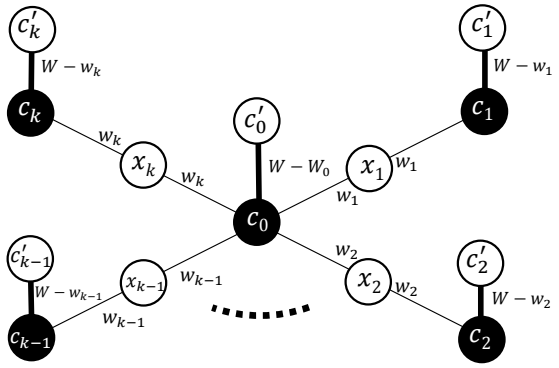

Figure 1: Gadget for NP-hardness of maximizing modularity on trees.

We show that an instance $I = (\{w_1, w_2, \ldots, w_k\};\ S)$ of SUBSET-SUM can be transformed into an instance $(T_I(V_I, E_I), c_I, K_I)$ of maximizing modularity on the tree T_I such that T_I has a community structure with modularity at least K_I if and only if there exists a subset of $\{w_1, w_2, \ldots, w_k\}$ that sums up to S. Since SUBSET-SUM is an NP-hard problem [24], it follows that there is no polynomial-time algorithm to decide the maximizing modularity problem on trees, unless $P = NP$.

The reduction is as follows: Given an instance $I = (\{w_1, w_2, \ldots, w_k\};\ S)$ of SUBSET-SUM, construct a tree $T_I = (V_I, E_I)$ consisting of $3k + 2$ vertices. For each integer $w_i \in I$, we introduce three vertices: a black vertex c_i and two white vertices c'_i and x_i. We also add a special pair: a black vertex c_0 and a white vertex c'_0. We connect each x_i to both c_i and c_0 with the weight w_i, and connect c'_i to c_i with weight $W - w_i$ for all $i = 1..n$, where $W = (2k + 1)\sum_i w_i$. The vertex c'_0 is connected to c_0 with weight $W - W_0$ where $W_0 = S + 1/2\sum_i w_i$. Before specifying parameters K_I, we characterize the important properties of any maximum modularity CS in a general graph $G = (V, E)$.

Lemma 1. *In a maximum modularity community structure of a graph $G = (V, E)$, the following properties hold.*

1. *There is no community with negative modularity [6].*
2. *Every non-isolated vertex is in the same community with at least one of its neighbors.*
3. *Each community induces a connected subgraph in G.*

Proof. Recall that the modularity of a single community X is defined as $Q(X) = E(X)/M - \mathsf{vol}(X)^2/4M^2$, where M is the sum of edge weights in the graph, $E(X)$ is the sum of weights of edges whose both endpoints are in X. The modularity of $\mathcal{C}$ equals the total modularity values of all communities in $\mathcal{C}$.

(1) The proof for the first property can be found in [6].

(2) Assume that there is a community $X \in \mathcal{C}$, a maximum modularity CS, that has only a single non-isolated vertex v. Since $\mathsf{vol}(X) = \mathsf{deg}(v) > 0$ and $E(X) = 0$, we have $Q(X) = E(X) - \mathsf{vol}(X)^2/4M^2 < 0$ that contradicts the first property.

(3) Assume that a community $X \in \mathcal{C}$ induces a disconnected subgraph in G. Thus X can be split into two disjoint subsets X_1 and X_2 such that no edges cross between X_1 and X_2. Therefore

$$\begin{aligned}\Delta Q &= Q(X_1) + Q(X_2) - Q(X) \\ &= E(X_1) + E(X_2) - E(X) + \frac{1}{4M^2}(\mathsf{vol}(X)^2 - \mathsf{vol}(X_1)^2 - \mathsf{vol}(X_2)^2) \\ &= \frac{1}{4M^2}((\mathsf{vol}(X_1) + \mathsf{vol}(X_2))^2 - \mathsf{vol}(X_1)^2 - \mathsf{vol}(X_2)^2) > 0,\end{aligned}$$

which contradicts the assumption that $\mathcal{C}$ has the maximum modularity. Hence every non-isolated vertex must be in the same community with at least one of its neighbors. □

Lemma 2. *In a maximum modularity community structure of T_I, each community has exactly one black vertex.*

Proof. Consider a maximum modularity CS $\mathcal{C}$. The proof consists of two parts: 1) There is no community of $\mathcal{C}$ that have all white nodes; and 2) There is no community in $\mathcal{C}$ with more than one black vertex.

For the first part, observe that none of the adjacent vertices have the same color. By the second property of Lemma 1, if a community contains a vertex u, it also contains a neighbor of u whose the color is different from u's. Thus every community must contain both black and white vertices.

We prove the second part by contradiction. Assume that a community $X \in \mathcal{C}$ has two black vertices. We show that $c_0 \in X$. Assume not, let c_i and c_j be two black vertices in X. Since $c_0 \notin X$, c_i and c_j are disconnected within the subgraph induced by X in T_I, contradicting the third property in Lemma 1. Hence, we assume w.l.o.g. that X contains c_0 and c_1. We prove that dividing X into two communities $X_1 = \{c_1, c_1'\}$ and $X_2 = X \setminus X_1$ increases the modularity, which contradicts the optimality of $\mathcal{C}$. That is to show

$$\Delta Q = -\frac{w_1}{M} + \frac{1}{4M^2}\left(\mathsf{vol}(X)^2 - \mathsf{vol}(X_1)^2 - \mathsf{vol}(X_2)^2\right) > 0, \tag{3}$$

where $M = (k+1)W + \sum_i w_i - W_0$ is the total weights of edges in T_I and for a given a subset $R \subseteq V$, $\mathsf{vol}(R)$ denotes the total weighted degree of the vertices in R.

Substitute $\mathsf{vol}(X) = \mathsf{vol}(X_1) + \mathsf{vol}(X_2)$ and simplify, we have

$$(3) \Leftrightarrow 4w_1M < 2\mathsf{vol}(X_1)\mathsf{vol}(X_2) \tag{4}$$

Since $w_1 < \sum_i w_i = \frac{1}{2k+1}W$, thus

$$4w_1M < \frac{2}{2k+1}W(k+1+\frac{1}{2k+1})W < 2W^2 \tag{5}$$

On the right-hand side of (4):

$$\begin{aligned}
\mathsf{vol}(X_2) &> 2W - \sum_i w_i = 2W - \frac{1}{2k+1}W \\
\mathsf{vol}(X_1) &> \mathsf{deg}(c_0') + \mathsf{deg}(c_0) = 2W - 2W_0 + \sum_i w_i = 2W - 2S \\
&> 2W - 2\sum_i w_i = 2W - \frac{2}{2k+1}W
\end{aligned}$$

When $k \geq 1$, we have

$$\begin{aligned}
2\mathsf{vol}(X_1)\mathsf{vol}(X_2) &> (2 - \frac{2}{2k+1})(2 - \frac{1}{2k+1})W^2 = (4 - \frac{6}{2k+1} + \frac{2}{(2k+1)^2})W^2 \\
&> 2W^2
\end{aligned} \tag{6}$$

It follows from (5) and (6) that (4) holds i.e. $\Delta Q > 0$. This contradicts the maximum modularity of $\mathcal{C}$. Thus each community in $\mathcal{C}$ must contain exactly one black vertex. □

Theorem 1. *Modularity maximization on trees is NP-complete.*

Proof. It is clear that maximizing modularity is in NP. To prove the NP-hardness, we reduce an instance $I = (\{w_1, w_2, \ldots, w_k\};\ S)$ of SUBSET-SUM to an instance $(T_I(V_I, E_I), c_I, K_I)$ of maximizing modularity on the tree T_I as presented.

Consider a maximum modularity CS $\mathcal{C}$. From Lemma 2 and the third property of Lemma 1, x_i is in the same community with either c_0 or c_i (but not both). Let $\delta_i = 1$ if x_i is in the same community with c_0 and $\delta_i = 0$, otherwise. In addition, let $S_0 = \sum_{\delta_i=1} w_i$ and $W_S = \sum_{i=1}^{k} w_i = \frac{1}{2k+1}W$.

For $1 \leq i \leq k$, exactly one of the two edges (x_i, c_0) or (x_i, c_i) will have two endpoints in two different communities. This leads to an important property that the total weights of edges whose endpoints belong to different community is exactly $\sum_{i=1}^{k} w_i$, we have:

$$\begin{aligned}
Q(\mathcal{C}) = \Big[1 - \frac{W_S}{M}\Big] - \frac{1}{4M^2}[&(2W - 2W_0 + W_S + 2S_0)^2 \\
&+ \sum_{\delta_i=0}(2W + w_i)^2 + \sum_{\delta_i=1}(2W - w_i)^2]
\end{aligned}$$

To maximize $Q(\mathcal{C})$, we need to minimize the second term that is

$$\begin{aligned}
&\frac{1}{4M^2}\Big[(2W-2W_0+W_S)^2+4S_0^2+4(2W-2W_0+W_S)S_0 \\
&\quad+\sum_{\delta_i=0}(4W^2+4Ww_i+w_i^2)+\sum_{\delta_i=1}(4W^2-4Ww_i+w_i^2)\Big] \\
=&\frac{1}{4M^2}\Big[(2W-2W_0+W_S)^2+4S_0^2+4(2W-2W_0+W_S)S_0 \\
&\quad+4kW^2+4WW_S-8WS_0+\sum_{i=1}^{k}w_i^2\Big] \\
=&\frac{1}{4M^2}\Big[(2W-2W_0+W_S)^2+4kW^2+4WW_S+\sum_{i=1}^{k}w_i^2) \\
&\quad+4\left(S_0^2-(2W_0-W_S)S_0\right)\Big].
\end{aligned}$$

For a fixed value of W_0, the above sum is minimized when $S_0^2-(2W_0-W_S)S_0$ is minimized at $S_0=\frac{2W_0-W_S}{2}=S$ (recall that we select $W_0=S+W_S/2$). Hence if we choose

$$K_I=\Big[1-\frac{W_S}{M}\Big]-\frac{1}{4M^2}\Big[(2W-2S)^2+4kW^2+4WW_S+\sum_{i=1}^{k}w_i^2)-4S^2\Big],$$

there is a CS in T_I with modularity at least K_I if and only if there is a subset of w_i (corresponding to $\delta_i=1$) that sum up to S. □

3.2 Polynomial Time Algorithm for Uniform-weighted Trees

We present a polynomial time algorithm for finding a maximum modularity CS on trees with uniform edge weights. By characterizing the structure of a maximum modularity CS, we reduce maximizing modularity on trees to the following problem.

Definition 3 (k-MSSV problem)**.** *Given a tree $T=(V,E)$, find a set of k edges that removal minimizes the sum-of-squares of component volumes in the residual forest.*

The relationship between maximizing modularity and k-MSSV is presented in the following lemma.

Lemma 3. *Let $\mathcal{C}$ be a maximum modularity CS of $T=(V,E)$, k be the number of communities in $\mathcal{C}$, and F be the set of edges whose two endpoints belong to two different communities. Then the following properties hold.*

1. *For any two different communities, there is at most one edge that crosses between them.*
2. $|F|=k-1$.

3. F is an optimal solution for $(k-1)$-MSSV problem on $T = (V, E)$.

Proof. By the first property in Lemma 1, each community induces a connected component. Thus we shall *use the terms community and component interchangeably* in the rest of this section.

(1) Assume that there are two different communities $C_1, C_2 \in \mathcal{C}$ and two different edges (u_1, v_1) and (u_2, v_2) that satisfy $u_1, u_2 \in C_1$ and $v_1, v_2 \in C_2$. By the first property in Lemma 1, there are paths between u_1 and u_2 within C_1 and between v_1 and v_2 within C_2. Those two paths together with the edges (u_1, v_1) and (u_2, v_2) form a cycle within the given tree T (contradiction).

(2) Abstract each community in $\mathcal{C}$ into a single node, we obtain a new graph T_A with k abstract vertices. The set of edges in the new graph are identical to the edges in F. Since T is connected and cycle-free, T_A is also connected and cycle-free. Thus T_A is a tree with k vertices. It follows that T_A has $k-1$ edges and so does F.

(3) By the second property, $Q(\mathcal{C}) = \left[1 - \frac{k-1}{|V|}\right] - \frac{1}{4|E|^2} \sum_{C_i \in \mathcal{C}} \mathsf{vol}(C_i)^2$. Thus $Q(\mathcal{C})$ is maximized when $\sum_{C_i \in \mathcal{C}} \mathsf{vol}(C_i)^2$ is minimized and vice versa. Hence F is an optimal solution for the $(k-1)$-MSSV problem on T. □

Thus a maximum modularity CS can be found by solving the k-MSSV problems with all k ranging from 0 to $|V|$. We introduce below a dynamic programming algorithm for maximizing modularity via solving the k-MSSV problem.

3.2.1 Dynamic Programming Algorithm

Given the tree $T = (V, E)$ with $|V| = n$, select a node $r \in V$ as the *root*. Denote by $T^u = (V^u, E^u)$ the subtree rooted at u in T with the set of vertices V^u and the set of edges E^u. Let $u_1, u_2, \ldots, u_{b(u)}$ denote the children of u, where $t(u) = \mathsf{deg}(u)$ if $u = r$ and $t(u) = \mathsf{deg}(u) - 1$ if $u \neq r$. In our dynamic programming algorithm, we define the following functions:

- $F^u(k)$: The minimum sum-of-squares of component volumes in T^u when k edges are removed. Note that even after removing k edges the component volumes are still measured as the sum of the vertex degrees in T, not T^u.
- $F^u(k, \nu)$: The minimum sum-of-squares of component volumes in T^u when k edges are removed and the component that contains u, called the *cap component*, has volume ν. In addition, if it is not possible to remove k edges to satisfy the two mentioned conditions, then $F^u(k, \nu) = \infty$
- $F_i^u(k, \nu)$: The minimum sum-of-squares of component volumes in $T_i^u = (V_i^u, E_i^u)$, the partial subtree formed by $u, T^{u_1}, T^{u_2}, \ldots, T^{u_i}$, when k edges are removed and the cap component has volume ν. As above, if it is not possible to remove k edges to satisfy the two conditions, then $F_i^u(k, \nu) = \infty$.

The maximum modularity value is given at the root r by

$$\max_{1\le k\le n-1, 1\le \nu\le 2(n-1)} \left\{ \frac{k}{n} - \frac{F^r(k,\nu)}{4(n-1)^2} \right\} \tag{7}$$

We compute $F^u(k,\nu)$ and $F_i^u(k,\nu)$ using the following recursions.

$$F^u(k) \quad = \min_{d_u \le \nu \le \mathsf{vol}(T^u)} F^u(k,\nu) \qquad \forall u \in V, k = 0..|E^u| \tag{8}$$

$$F^u(k,\nu) = F^u_{t(u)}(k,\nu) \qquad \forall u \in V, k = 0..|E^u|, \nu = d_u..\mathsf{vol}(T^u) \tag{9}$$

$$F_i^u(k,\nu) = \min \begin{cases} \min\limits_{0\le l\le k-1} F^u_{i-1}(l,\nu) + F^{u_i}(k-l-1) & \text{(10.a)} \\ \min\limits_{0\le l\le k, 0\le \mu\le \nu} F^u_{i-1}(l,\mu) + F_i^{u_i}(k-l,\nu-\mu) + 2\mu(\nu-\mu) & \text{(10.b)} \end{cases}$$
$$\forall u \in V, k = 1..|E_i^u|, i = 1..t(u), \nu = 1..\mathsf{vol}(T^u) \tag{10}$$

The basis cases are as follows.

$$F_i^u(0,\nu) = \begin{cases} \mathsf{vol}(T_i^u)^2 & \text{if } \nu = \mathsf{vol}(T_i^u) \\ \infty & \text{otherwise} \end{cases} \tag{11}$$
$$\forall u \in V, k = 1..|E_i^u|, i = 1..t(u), \nu = 0..\mathsf{vol}(T^u)$$

$$F_0^u(k,\nu) = \begin{cases} d_u^2 & \text{if } \nu = d_u \\ \infty & \text{otherwise} \end{cases} \tag{12}$$
$$\forall u \in V, k = 0..|E_i^u|, \nu = 0..\mathsf{vol}(T^u)$$

$$F_i^u(k,\nu) = \infty \qquad \forall u \in V, k = 0..|E_i^u|, i = 0..t(u), \nu < d_u \tag{13}$$

We explain the recursion from (8) to (10). In (8), we consider all possible ν, the volume of the cap component, and assign $F^u(k)$ the minimum values among $F^u(k,\nu)$. The definition of $F^u(k,\nu)$ is straightforward as shown in (9). Finally, we compute $F_i^u(k,\nu)$ in (10) as the minimum of the following two cases:

- If (u,u_i) is removed (10.a): We can remove at most $k-1$ other edges. In addition, the cap component in T_i^u has the same volume ν with that of T_{i-1}^u. For $l = 0..k$, we take the sum of the optimal solution when removing l edges in T_{i-1}^u and the optimal solution when removing $k-l-1$ edges in T_{u_i} and take the minimum as a possible value for $T_i^u(k,\nu)$.
- If (u,u_i) is not removed (10.b): We can remove l edges within T_{i-1}^u and $k-l$ removed edges within T^{u_i} for some $0 \le l \le k$. In addition, if the cap component volume of T_{i-1}^u is μ, then the cap component of T^{u_i} has volume $\nu-\mu$. The factor $2\mu(\nu-\mu) = \nu^2 - \mu^2 - (\nu-\mu)^2$ accounts for the increment of the total sum-of-squares volumes.

The running time of the dynamic algorithm is stated in the following lemma.

Lemma 4. *The recursions from (8) to (10) can be computed in $O(n^5)$.*

Proof. There are at most n different sets of $F_i^u(\cdot)$ to compute. For each set $F_i^u(\cdot)$, there are at most $n \times 2n$ pairs of (k,ν) and the time to compute each $F_i^u(k,\nu)$ is at most $n \times 2n$ (Eq. 10). Thus the running time is bounded by $O(n^5)$. □

We summarize the dynamic programming algorithm to maximize modularity in the following algorithm (Algorithm 1).

Algorithm 1. Dynamic Programming Algorithm for Uniform-weight Trees

1. **for** $u \in V$ in postorder traversal from r **do**
2. **for** $i = 0$ to $t(u)$ **do**
3. **for** $k = 0$ to $|E(u)|$ **do**
4. **for** $\nu = 0$ to $\mathsf{vol}(T^u)$ **do**
5. Compute $F_i^u(k,\nu), F^u(k,\nu)$, and $F^u(k)$ using (8)-(13).
6. $Q = 0$
7. **for** $k = 0$ to n **do**
8. **for** $\nu = 0$ to $\mathsf{vol}(T)$ **do**
9. $Q = \max\{Q, \frac{k}{n} - \frac{F^r(k,\nu)}{4(n-1)^2}\}$
10. return Q

Theorem 2. *Algorithm 1 finds the maximum modularity in uniform weights trees in $O(n^5)$.*

3.3 Pseudo-Polynomial Time Algorithm

The above dynamic algorithm can be generalized to work for *non-uniform integral weight* trees. The recursion is similar to (8)-(10) with the differences in the bounds for k and ν.

$$F^u(k) = \min_{d_u \le \nu \le \mathsf{vol}(T^u)} F^u(k,\nu) \quad \forall u \in V, k = 0..\mathsf{vol}(T^u)/2 \tag{14}$$

$$F^u(k,\nu) = F^u_{t(u)}(k,\nu) \quad \forall u \in V, k = 0..W^u, \nu = 0..\mathsf{vol}(T^u) \tag{15}$$

$$F_i^u(k,\nu) = \min \begin{cases} \min_{0 \le l \le k} F^u_{i-1}(l,\nu) + F^{u_i}(k-l) & \text{(16.a)} \\ \min_{0 \le l \le k-1, 0 \le \mu \le \nu} F^u_{i-1}(l,\mu) & \\ \qquad + F_i^{u_i}(k-l-w(u,u_i), \nu-\mu) + 2\mu(\nu-\mu) & \text{(16.b)} \end{cases} \tag{16}$$

$$\forall u \in V, k = 0..\mathsf{vol}(T^u), i = 1..t(u), \nu = 0..\mathsf{vol}(T_i^u). \tag{17}$$

The time complexity of the algorithm is a function of $vol(T)$, which can be exponentially large in terms of the input size. Thus the dynamic programming is a pseudo-polynomial time algorithm with the time complexity stated in the following theorem.

Theorem 3. *The recursion (14)-(17) gives an $O(n^5W^4)$ pseudo-polynomial time algorithm for maximizing modularity on integral weight tree, where W is the maximum edge weight.*

Corollary 1. *Modularity maximization on weighted trees is weakly NP-complete.*

3.4 Polynomial Time Approximation Scheme (PTAS)

Given an $\epsilon > 0$, we present an $O(n^{1/\epsilon+1})$ algorithm to find CS in T with modularity at least $(1-\epsilon)Q_{\mathsf{opt}}$, where Q_{opt} denotes the maximum modularity over all possible CS.

On one hand, the second properties in Lemma 3 states that a CS with k communities has exactly $k-1$ edges whose endpoints in different communities. Thus there is a one-on-one correspondence between a set of $k-1$ edges in T and CS with exactly k communities: removing $k-1$ edges yields k connected components/communities. On the other hand, the following lemma implies that CS with at most k communities approximate closely to the maximum modularity.

Lemma 5. *[16, 21] Given a weighted graph $G = (V, E)$. Denote by Q_k the maximum modularity of a CS in G with* at most *k communities and by $Q_{\text{opt}} = Q_n$. We have*

$$Q_k \geq (1 - \frac{1}{k})Q_{\text{opt}}$$

Thus we have the following PTAS for maximizing modularity on trees.

Algorithm 2. PTAS for Maximizing modularity on Trees

1. Given $\epsilon > 0$, set $k = \lceil 1/\epsilon \rceil$.
2. $Q_k = 0, \mathcal{C}_k = \{V\}$
3. **for each** $X \subset E$ and $|X| < k$ **do**
4. Find connected component $C_1, C_2, \ldots, C_k$ in $T' = (V, E \setminus X)$.
5. Let CS $\mathcal{J} = \{C_1, C_2, \ldots, C_k\}$
6. **if** $Q(\mathcal{J}) > Q_k$ **then**
7. $Q_k = Q(\mathcal{J})$
8. $\mathcal{C}_k = \mathcal{J}$
9. Return $\mathcal{C}_k$

Theorem 4. *Algorithm 2 finds in $O(n^{1+1/\epsilon})$ time a CS with modularity value at least $(1-\epsilon)Q_{opt}$ on weighted trees.*

Proof. By Lemma 5, the modularity of the found CS will be at least $(1-\epsilon)Q_{\mathsf{opt}}$. In addition, for each of the n^{k-1} subsets, computing the modularity takes $O(n)$. Thus the total time complexity is $O(n^{k-1+1}) = O(n^{1/\epsilon+1})$. □

While the existence of an FPTAS implies the existence of a pseudo-polynomial time algorithm, the reverse is not necessary true. It is an open question that whether an FPTAS for maximizing modularity on trees exists.

4 Linear Programming Based Algorithm

In this section, we first present the original linear program (LP) in [15]. Then we present, in subsection 4.2, our *sparse metric* formulations that contain only a small fraction of constraints in the original LP. We postpone all the proofs on correctness and performance of our new formulations till the end of the section.

4.1 The Linear Program and The Rounding

The modularity maximization problem can be formulated as an Integer Linear Programming (ILP) [14] [2].(18) to (21), has one variable x_{ij} for each pair (i, j) of vertices to represent the "distance" between i and j i.e. $x_{ij} = 0$ if i and j are in the same community and $x_{ij} = 1$, otherwise.

In other words, x_{ij} is equivalent to $1-\delta_{ij}$ in the definition (1) of modularity. Thus, the objective function to be maximized can be written as $\sum_{ij} B_{ij}(1-x_{ij})$.

The ILP to maximize modularity ($\text{IP}_{complete}$) is as follows

$$\text{maximize} \quad \frac{1}{2M}\sum_{ij} B_{ij}(1-x_{ij}) \tag{18}$$

$$\text{subject to} \quad x_{ij} + x_{jk} - x_{ik} \geq 0, \qquad \forall i<j<k \tag{19}$$

$$x_{ij} - x_{jk} + x_{ik} \geq 0, \qquad \forall i<j<k \tag{20}$$

$$-x_{ij} + x_{jk} + x_{ik} \geq 0, \qquad \forall i<j<k \tag{21}$$

$$x_{ij} \in [0,1], \qquad i,j \in [1..n], \tag{22}$$

Constraints (19), (20), and (21) are well-known *triangle inequalities* that guarantee the values of x_{ij} are consistent to each other. They imply the following transitivity: if i and j are in the same community and j and k are in the same community, then so are i and k. By definition, $x_{ii} = 0\ \forall i$ and can be removed from the ILP for simplification.

We shall refer to the IP described above as $\text{IP}_{complete}$ and its relaxation, obtained by replacing the constraints $x_{ij} \in \{0,1\}$ by $x_{ij} \in [0,1]$, as $\text{LP}_{complete}$. If the optimal solution of this relaxation is an integral solution, which is very often the case [25], we have a partition with the maximum modularity. Otherwise, we resort on rounding the fractional solution and use the value of the objective as

[2] In deed, the constraints in [14] are of the form $x_{ij} + x_{jk} - 2x_{ik} \leq 1$ rather than the form in Eq. 19. As a result, the LP relaxation in [14] is less tight than those in [15] and Eq. 19-21.

an upper-bound that enables us to lower-bound the gap between the rounded solution and the optimal integral solution.

Agarwal and Kempe [15] used a simple rounding algorithm, proposed by Charikar et al. [26] for the *correlation clustering* problem [27], to obtain the community structure from a fractional optimal solution. The values of x_{ij} are interpreted as a metric "distance" between vertices. The algorithm repeatedly groups all vertices that are close by to a vertex into a community. The final community structure are then refined by a Kernighan-Lin [28] local search method.

The modularity maximization formulation can be also expressed as a clique partitioning problem [29]. In [29] the author proposed a row generation technique to incrementally add the triangle inequalities constraints and solve the LP. In each iteration, the batch of about 150 constraints are added and the non-tight constraints are identified and removed from the LP. The advantage of our sparse LP over the row generation method is that, the sparse metric technique exclude major 'redundant' constraints even before solving the programming formulation. Since the set of non-tight constraints are known as *a priori*, the LP is solved only once and we do not have to examine the $O(n^3)$ constraints in each iteration. Nevertheless, the two techniques are orthogonal and can be applied in parallel to improve the running time. Regarding efforts on solving the IP exactly, cutting planes and polyhedral characteristics for the clique partitioning and other clustering problems can be found in [25,30,31] and the references therein.

Since the rounding phase is comparatively simple, the burden of both time and memory comes from solving the large LP relaxation. Note that the LP has $\binom{n}{2}$ variables and $3\binom{n}{3} = \Theta(n^3)$ constraints that is about half a million constraints for a network of 100 vertices. As a consequence, the sizes of solved instances in [15] were limited to few hundred nodes. Hence, there is a need for more efficient formulations for the maximizing modularity problem. By combining mathematical approaches with combinatorial techniques, we achieve this goal in next subsection.

4.2 Sparse Metric

Instead of using $3\binom{n}{3}$ triangle inequalities, we show that only a small subset of those, called *sparse metric*, are sufficient to obtain the same optimal solutions.

Our integer linear program with the *Sparse Metric* technique, denoted by IP_{sparse}, is as follows:

$$\text{maximize} \qquad -\frac{1}{2M}\sum_{ij} B_{ij}x_{ij} \tag{23}$$

$$\text{subject to} \qquad x_{ik} + x_{kj} \geq x_{ij} \qquad k \in K(i,j) \subseteq V \setminus \{i,j\} \tag{24}$$

$$x_{ij} \in \{0,1\}, \tag{25}$$

First, since $\sum_{ij} B_{ij} = \sum_{ij} A_{ij} - \frac{d_i d_j}{2M} = 2M - \frac{\sum_i d_i \sum_j d_j}{2M} = 0$, we simplify

the objective to $-\frac{1}{2M}\sum_{ij} B_{ij}x_{ij}$. Second, different selections of $K(i,j)$ give us different formulations. For example, the $\text{IP}_{complete}$ can be obtained by choosing $K(i,j) = V \setminus \{i,j\}$. However, not all selection of $K(i,j)$ result in valid formulations for the maximizing modularity problem.

Notice that if we define a function $d(i,j) = x_{ij}$, then the function should satisfy all the following conditions of a pseudo-metric:

1. $d(i,j) \geq 0$ (non-negativity)
2. $d(i,i) = 0$ (and possibly $d(i,j) = 0$ for some distinct values $i \neq j$)
3. $d(i,j) = d(j,i)$ (symmetry)
4. $d(i,j) \leq d(i,k) + d(k,j)$ (transitivity).

Therefore, $K(i,j)$ must be selected so that if $x \in [0,1]^{\binom{n}{2}}$ is a feasible solution then x must induce a pseudo-metric.

We will prove in the next section (Theorems 5 and 6) that if $K(i,j)$ is a vertex cut [3] for two vertices i and j, then IP_{sparse} is a valid formulation for the maximizing modularity problem. Moreover, the corresponding LP relaxation, called LP_{sparse}, will have the same strength with the $\text{LP}_{complete}$ i.e. they have the same optimal objective values.

Hence, we select $K(i,j)$ as the minimum set of vertices whose deletion disconnects i from j. The cardinality of $K(i,j)$ is known as the vertex connectivity of i and j and is denoted by $\kappa(i,j)$. Alternatively, we can select $K(i,j)$ to be neighbors of i or j which are much easier to find. Either ways will give the following bounds on the number of constraints.

Lemma 6. *If $d_1 \leq d_2 \leq \ldots \leq d_n$ is the sorted (unweighted) degree sequence of the graph, then the number of constraints is upper bounded by the following quantities*

$$\textit{1)} \sum_{i=1}^{n}(i-1)d_i \qquad \textit{2)}\ m(n-1)$$

where $m = |E|$ is the number of edges.

Proof. Since $|K(i,j)| \leq \min\{d_i, d_j\}$, hence the number of constraints is at most $\sum_{i<j}\min\{d_i,d_j\} = \sum_{i<j} d_i = \sum_{i=1}^{n}(i-1)d_i$. Also, since $\min\{d_i,d_j\} \leq 1/2(d_i + d_j)$, thus the number of constraints is upper-bounded by

$$\sum_{i<j}\frac{1}{2}(d_i+d_j) = \frac{1}{2}\sum_{i=1}^{n}(n-1)d_i = m(n-1)$$

This completes the proof. □

[3] a set of vertices whose removal from the graph disconnects i and j. If $(i,j) \in E$, $K(i,j)$ is a vertex cut of i and j after removing edge (i,j).

Corollary 2. *In sparse networks, where $m = O(n)$, LP_{sparse} contains only $O(n^2)$ constraints.*

Thus, for sparse networks, our new formulations substantially reduces time and memory requirements. For most real-world network instances, where $n \approx m$, the number of constraints is effectively reduced from $\Theta(n^3)$ to $O(n^2)$. If we consider the time to solve a linear program to be cubic time the number of constraints, the total time complexity for sparse networks improves to $O(n^6)$ instead of $O(n^9)$ as in the original approach.

4.3 Validity and Strength of the Sparse Formulations

We show the equivalence between the sparse formulation and the complete formulation when $K(i, j)$ is selected as a *vertex cut* of i and j. We prove in Theorems 5 and 6 the following statements, respectively.

- IP_{sparse} and $\text{IP}_{complete}$ have the same set of optimal integral solutions.
- The optimal fractional solutions of LP_{sparse} and $\text{LP}_{complete}$ have the same objective values i.e. they provide the same upper bound on the maximum possible modularity.

Hence, solving LP_{sparse} indeed gives us an optimal solution of $\text{LP}_{complete}$ within only a small fraction of time and memory requirements of $\text{LP}_{complete}$.

Theorem 5. *Two integer programmings IP_{sparse} and $IP_{complete}$ have the same set of optimal solutions.*

Proof. We need to show that every optimal solution of $\text{IP}_{complete}$ is also a solution of IP_{sparse} and vice versa. In one direction, since the constraints in IP_{sparse} are a subset of constraints in $\text{IP}_{complete}$, every optimal solution of $\text{IP}_{complete}$ will also be a solution of IP_{sparse}.

In the other direction, let x_{ij} be an optimal integral solution of IP_{sparse}. We shall prove that x_{ij} must be a pseudo-metric that implies x_{ij} is also a feasible solution of $\text{IP}_{complete}$. For convenience, we assume that the original graph $G = (V, E)$ has no isolate vertices that are known to have no affects on modularity maximization [23]. Construct a graph $G_d = (V, E_d)$ in which there is an edge $(i, j) \in E_d$ for every $x_{ij} = 0$. Let $\mathcal{C}_d = \{C_d^1, C_d^2, \ldots, C_d^l\}$ be the set of connected components in G_d, where C_d^t is the set of vertices in tth connected component.

We first prove an important property in the optimal community structure. That is each community must induce a connected subgraph in the network.

Claim: *Every connected component C_d^t induces a* connected *subgraph in the graph $G = (V, E)$.*

Proof. We prove by contradiction. Assume that the connected component C_d^t does not induce a connected subgraph in G. Hence, we can partition C_d^t into two subsets S and T so that there are no edges between S and T in G.

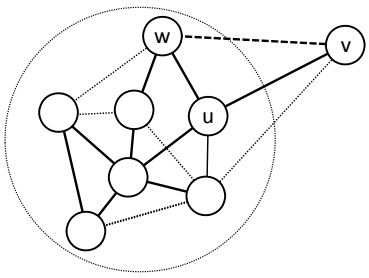

Figure 2: Clique expanding process.

Construct a new solution x' from x by setting $x'_{ij} = 1$ for every pair $(i,j) \in S \times T$ and $x'_{ij} = x_{ij}$ otherwise. For every pair $(i,j) \in S \times T$, since $A_{ij} = 0$, we have $B_{ij} = A_{ij} - \frac{d_i d_j}{2M} < 0$. Hence, setting $x'_{ij} = 1 \ \forall (i,j) \in S \times T$ can only increase the objective value. In fact, since there must be at least one pair $(i,j) \in S \times T$ with $x_{ij} = 0$ (or else C_d^t is not a connected component in G_d), doing so will strictly increase the objective. Moreover, we can verify that x' also satisfy all constraints of IP_{sparse}. Thus, we have a feasible solution with higher objective than the optimal solution x (contradiction). □

To prove that x_{ij} is a pseudo-metric, we prove an equivalent statement that if vertices i and j belong to the same component C_d^t then the distance $x_{ij} = 0$. We prove by repeatedly growing a clique inside C_d^t. At each step, every pair of vertices in the clique are proven to have distance 0. Then, we add one more vertex to the clique and prove that the new vertex is also of distance zero from other vertices in the clique (see Fig. 2).

Formally, we prove by induction that for all $p \leq |C_d^t|$, there exists in C_d^t a clique Ψ of size p satisfying simultaneously the following conditions

1. $x_{ij} = 0 \ \forall (i,j) \in \Psi$,
2. The subgraph induced by Ψ in G is connected.

The basis. For $p = 1$, select an arbitrary vertex in C_d^t. The two above conditions hold trivially.

The inductive step. Assume that we have a clique $\Psi \subset C_d^t$ satisfying the two conditions. If $\Psi = C_d^t$ then we have completed the proof. Otherwise, there exist vertex $u \in \Psi$ and vertex $v \in C_d^t \setminus \Psi$, so that $(u,v) \in E(G)$ and $x_{uv} = 0$. The existence of such an edge (u,v) can be proven by contradiction. Assume not, then we can increase distance of all pairs in $\Psi \times (C_d^t \setminus \Psi)$ from 0 to 1 to increase the objective value while not violating any constraints. This would imply Ψ is disconnected from the rest of C_d^t, which is contradicted to the fact that C_d^t induces a connected subgraph. Now, consider an arbitrary neighbor w of u in the subgraph induced by Ψ. Since u is a common neighbor of w and v, we have $u \in K(w,v)$ i.e. the constraint $x_{wu} + x_{uv} \geq x_{wv}$ must be in IP_{sparse}. Thus, we have $x_{wv} = 0$ since both $x_{wu} = 0$ $(w, u \in \Psi)$ and $x_{uv} = 0$.

We have proven that for every neighbor w of u, $x_{wv} = 0$. Similarly, for all neighbor w' of w, $x_{vw'} = 0$, and so on. Since the clique induces a connected subgraph in G, eventually $\forall u' \in \Psi$, we have $x_{vu'} = 0$. That is we can extend the clique Ψ to include v and obtain a clique of size $p+1$ that satisfies the two conditions. This completes the proof of Theorem 5. □

Theorem 6. *LP_{sparse} and $LP_{complete}$ share the set of optimal solutions which are extreme points.*

Proof. We need to show that every *fractional optimal* solution of $\text{LP}_{complete}$ is also a *fractional* solution of LP_{sparse} and vice versa. Since the integrality constraints have been dropped in both LP relaxations, we need a different approach to the proof in Theorem 5.

First, every fractional optimal solution of $\text{LP}_{complete}$ is also a *fractional* solution of LP_{sparse}. For the other direction, let x be a fractional optimal solution of LP_{sparse}, we shall prove that x is also a feasible solution of $\text{LP}_{complete}$.

Associate a weight $w_{ij} = x_{ij}$ for each edge $(i,j) \in E(G)$ (other edges are assigned weights ∞). Let x'_{ij} be the shortest distance between two vertices i and j within $G = (V, E)$ given the new edge weights. We shall prove the following statements

1. $x'_{ij} = \min_{k=1}^{n}\{x'_{ik} + x'_{kj}\}$.
2. $x'_{ij} \geq x_{ij}$ for all i, j and $x'_{ij} = x_{ij}$ $\forall (i,j) \in E$.

The first statement is obvious by the definition of x'_{ij}. We prove the second statement by contradiction. Assume that there exist i and j such that $x'_{ij} < x_{ij}$. Let $u_0 = i, u_1, \ldots, u_l = j$ be the vertices on the shortest path between i and j. We also assume that among pairs of vertices (i,j) satisfying $x'_{ij} < x_{ij}$, we select the pair with minimum value of l, the number of edges on the shortest path.

Since $K(i,j)$ is a vertex cut of i and j, there must be $0 < k < l$ such that $u_k \in K(i,j)$ i.e. the constraint $x_{iu_k} + x_{u_kj} \geq x_{ij} > x'_{ij}$ appears in the LP_{sparse}. From the suboptimality of the shortest path, we have $x'_{iu_k} = x_{u_0u_1} + \ldots + x_{u_{k-1}u_k}$ and $x'_{u_kj} = x_{u_ku_{k+1}} + \ldots + x_{u_{l-1}u_l}$ and $x'_{ij} = x'_{iu_k} + x'_{u_kj}$. Therefore

$$x_{iu_k} + x_{u_kj} \geq x_{ij} > x'_{ij} = x'_{iu_k} + x'_{u_kj}$$

That is either $x_{iu_k} > x'_{iu_k}$ or $x_{u_kj} > x'_{u_kj}$. However, the length of the shortest paths between i and u_k and between u_k and j are strictly less than l. We obtain the contradiction to the minimal selection of l and complete the proof of the second statement.

The two statements imply that x'_{ij} is a pseudo-metric. However, x'_{ij} may be no longer upper bounded by one. Thus, we define $x^*_{ij} = \min\{x'_{ij}, 1\}$ that satisfy the following properties.

- $x^*_{ij} \geq x_{ij}$ $\forall i, j$ (by definition),
- $x^*_{ij} = x'_{ij} = x_{ij}$ $\forall (i,j) \in E$, and

- $x^*_{ik} + x^*_{kj} \geq \min\{x'_{ik} + x'_{kj}, 1\} \geq \min\{x'_{ij}, 1\} = x^*_{ij}$.

That is x^* is also a pseudo-metric.

Now, if $x_{ij} = x^*_{ij}$ for all i, j, then x satisfies all triangle inequalities in $\text{LP}_{complete}$ and we yield the proof. Otherwise, assume that $x_{ij} < x^*_{ij}$ for some pair (i, j). We show that x^* is a feasible solution of LP_{sparse} with a greater objective value that contradicts the hypothesis that x is an optimal solution. Indeed, for all edges $(i, j) \notin E(G)$, $x_{ij} = x^*_{ij}$, and for $(i, j) \notin E$ we have $(B_{ij} < 0) \wedge (x^*_{ij} \geq x_{ij})$. Hence, the objective $-\frac{1}{2M}\sum_{ij} B_{ij}x^*_{ij} > -\frac{1}{2M}\sum_{ij} B_{ij}x_{ij}$ (contradiction). □

Table 1: Order and size of network instances

Problem ID	Name	Vertices (n)	Edges (m)
1	Zachary's karate club	34	78
2	Dolphin's social network	62	159
3	Les Miserables	77	254
4	Books about US politics	105	441
5	American College Football	115	613
6	US Airport 97	332	2126
7	Electronic Circuit (s838)	512	819
8	Scientific Collaboration	1589	2742

5 Computational experiments

We compare the running time and the number of constraints of our sparse metric formulations and the original LP in [15]. In addition, we include in the comparison the modularity values of the most popular algorithms in the literature [13, 15, 23]. Also, we include the state of the art, the Blondel method, [20]. Since Blondel is a randomized algorithm, we repeat the algorithm 10 times and report the best modularity value found. The optimal modularity values are reported in [25] except for the largest test case in which we use the GUROBI built-in branch-and-cut algorithm to find the optimal integral solution.

We perform the experiments on the standard datasets for community structure identification [15, 25], consisting of real-world networks. The datasets' names together with their sizes are listed in Table 1. The largest network consists of 1580 vertices and 2742 edges. All references on the datasets can be found in [15] and [25]. The LP solver is GUROBI 4.5, running on a PC computer with Intel 2.93 Ghz processor and 12 GB of RAM.

Since the same rounding procedures are applied on the optimal fractional solutions, both $\text{LP}_{\text{complete}}$ and $\text{LP}_{\text{sparse}}$ yield the same modularity values. However, $\text{LP}_{\text{sparse}}$ can run on much larger network instances. The modularity of the rounding LP algorithms and other published methods are shown in Table 2.

Table 2: The modularity obtained by previous published methods GN [13], EIG [23], Blondel [20], VP [15], $\mathrm{LP}_{\mathrm{cp}}$ (complete LP) [15], our *sparse metric* approach $\mathrm{LP}_{\mathrm{sp}}$ and the optimal modularity values OPT [25].

ID	n	GN	EIG	Blondel	VP	$\mathrm{LP}_{\mathrm{cl}}$	$\mathrm{LP}_{\mathrm{sp}}$	OPT
1	34	0.401	0.419	0.420	0.420	0.420	0.420	0.420
2	62	0.520	-	0.524	0.526	0.529	0.529	0.529
3	77	0.540	-	0.560	0.560	0.560	0.560	0.560
4	105	-	0.526	0.527	0.527	0.527	0.527	0.527
5	115	0.601	-	0.605	0.605	0.605	0.605	0.605
6	332	-	-	-	-	-	0.368	0.368
7	512	-	-	0.796	-	-	0.819	0.819
8	1589	-	-	0.955	-	-	0.955	0.955

The rounding LP algorithm can find optimal solutions (or within 0.1% of the optimal solutions) in all cases. While getting optimal modularity values with exact algorithms is cost-prohibitive, rounding fractional solutions of our LP_{sparse} takes less than 2 minutes for moderate size networks.

Note that only our rounding LP method can work on the test case 6, where the tested network is *directed*. The reason is that popular modularity optimization methods such as GN, EIG, and Blondel cannot work with directed networks; and the previous LP formulation [15] is too large to fit into the memory.

Table 3: Number of constraints and running time in seconds of the formulations $\mathrm{LP}_{\mathrm{complete}}$ and LP_{sparse}. $\langle$C$\rangle$ stands for *complete* and $\langle$S$\rangle$ stands for *sparse*.

ID	n	Constraint$\langle$C$\rangle$	Constraint$\langle$S$\rangle$	Time$\langle$C$\rangle$	Time$\langle$S$\rangle$
1	34	17,952	1,441	0.21	0.02
2	62	113,460	5,743	3.85	0.11
3	77	219,450	6,415	13.43	0.08
4	105	562,380	30,236	60.40	1.76
5	115	740,715	66,452	106.27	13.98
6	332	18,297,018	226,523	-	197.03
7	512	66,716,160	294,020	-	53.18
8	1589	2,002,263,942	159,423	-	2.94

Finally, we compare the number of constraints of the LP formulation used in [15] and our new formulation ($\mathrm{LP}_{\mathrm{sparse}}$) in Table 3. Our new formulation contains substantially less constraints, thus can be solved more effectively. The old LP formulation cannot be solved within the time allowance (10000 seconds) and the memory availability (12 GB) in cases of the network instances 6 to 8. The largest instance of 1589 nodes is solved surprisingly fast, taking under 3 seconds. The reason is due to the presence of leaves (nodes of degree one)

and other special motifs that can be efficiently preprocessed with the reduction techniques in [32].

Our new technique substantially reduces the time and memory requirements both theoretically and experimentally without any trade-off on the quality of the solution. The size of solved network instances raises from hundred to several thousand nodes while the running time on the medium-instances are sped up from 10 to 150 times. Thus, the *sparse metric* technique is a suitable choice when the network has a moderate size and a community structure with performance guarantees is desired.

References

[1] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393, 1998.

[2] A. Barabasi, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A*, 281, 2000.

[3] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science (New York, N.Y.)*, 298(5594), 2002.

[4] S. Fortunato and C. Castellano. Community structure in graphs. *Encyclopedia of Complexity and Systems Science*, 2008.

[5] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. Botgraph: large scale spamming botnet detection. In *NSDI '09*, pages 321–334. USENIX Association, 2009.

[6] T.N. Dinh, Ying Xuan, and M.T. Thai. Towards social-aware routing in dynamic communication networks. In *Proc. of IEEE IPCCC*, pages 161–168, 2009.

[7] N.P. Nguyen, T.N. Dinh, Y. Xuan, and M.T. Thai. Adaptive algorithms for detecting community structure in dynamic social networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 2282 –2290, april 2011.

[8] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 10(11):1576 –1589, nov. 2011.

[9] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Proceedings of the ACM SIGCOMM 2006 conference*, SIGCOMM '06, pages 267–278, New York, NY, USA, 2006. ACM.

[10] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An analysis of social network-based sybil defenses. In *Proceedings of the ACM SIGCOMM*

2010 conference, SIGCOMM '10, pages 363–374, New York, NY, USA, 2010. ACM.

[11] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci. A social network based patching scheme for worm containment in cellular networks. In *INFOCOM 2009, IEEE*, pages 1476 –1484, april 2009.

[12] B. Pásztor, L. Mottola, C. Mascolo, G.P. Picco, S. Ellwood, and D. Macdonald. Selective reprogramming of mobile sensor networks through social community detection. In *Proc. of EWSN*, volume 5970, pages 178–193. Springer Berlin Heidelberg, 2010.

[13] M. Girvan and M. E. Newman. Community structure in social and biological networks. *PNAS*, 99(12), 2002.

[14] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 20(2), 2008.

[15] G. Agarwal and D. Kempe. Modularity-maximizing graph communities via mathematical programming. *Eur. Phys. J. B*, 66, 2008.

[16] Bhaskar DasGupta and Devendra Desai. On the complexity of newman's community finding approach for biological and social networks. *Journal of Computer and System Sciences*, (0), 2012.

[17] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1), 2007.

[18] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80:056117, Nov 2009.

[19] Benjamin H. Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81:046106, Apr 2010.

[20] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008.

[21] T. N. Dinh and M. T. Thai. Finding community structure with performance guarantees in scale-free networks. In *SocialCom/PASSAT*, pages 888–891, 2011.

[22] T.N. Dinh and M.T. Thai. Precise structural vulnerability assessment via mathematical programming. In *IEEE Military Conference - MILCOM 2011*, pages 1351 –1356, 2011.

[23] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103, 2006.

[24] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[25] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron, and L. Liberti. Column generation algorithms for exact modularity maximization in networks. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 82, 2010.

[26] M. Charikar and A. Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. *FOCS*, 2004.

[27] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Foundations of Computer Science, Annual IEEE Symposium on (FOCS)*, 0:238, 2002.

[28] B. W. Kemighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Journal of Classification*, 1970.

[29] M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Math. Program.*, 45(1):59–96, August 1989.

[30] M. Grötschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming*, 47(1):367–387, 1990.

[31] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79:191–215, 1997. 10.1007/BF02614317.

[32] Duch J Fernandez A Gomez S Arenas, A. Size reduction of complex networks preserving modularity. *New J. Phys.*, 9, 2007.