Theoretical Computer Science 593 (2015) 117–131

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Algorithmic aspects of homophyly of networks

Peng Zhang [a,*], Angsheng Li [b]

[a] School of Computer Science and Technology, Shandong University, Jinan 250101, China
[b] State Key Lab. of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 4 November 2012
Accepted 2 June 2015
Available online 9 June 2015
Communicated by G. Ausiello

Keywords:
Homophyly
Social networks
Maximum happy vertices
Maximum happy edges
Approximation algorithms

ABSTRACT

We investigate the algorithmic problems of the *homophyly phenomenon* in networks. Given an undirected graph $G = (V, E)$ and a vertex coloring $c: V \to \{1, 2, \cdots, k\}$ of G, we say that a vertex $v \in V$ is *happy* if v shares the same color with all its neighbors, and *unhappy*, otherwise, and that an edge $e \in E$ is *happy*, if its two endpoints have the same color, and *unhappy*, otherwise. Supposing c is a *partial vertex coloring* of G, we define the Maximum Happy Vertices problem (MHV, for short) as to color all the remaining vertices such that the number of happy vertices is maximized, and the Maximum Happy Edges problem (MHE, for short) as to color all the remaining vertices such that the number of happy edges is maximized.

Let k be the number of colors allowed in the problems. We show that both MHV and MHE can be solved in polynomial time if $k = 2$, and that both MHV and MHE are NP-hard if $k \geq 3$. We devise a $\max\{1/k, \Omega(\Delta^{-3})\}$-approximation algorithm for the MHV problem, where Δ is the maximum degree of vertices in the input graph, and a 1/2-approximation algorithm for the MHE problem. This is the first theoretical progress of these two natural and fundamental new problems.

1. Introduction

Networks or at least social networks heavily depend on human or social behaviors. It is believed that *homophyly* [4, Chapter 4] is one of the most basic notions governing the structure of social networks. It is a common sense principle that people are more likely to connect with people they like, as what says in the proverb "birds of a feather flock together".

Li and Peng in [14,15] gave a mathematical definition of *community*, and *small community phenomenon* of networks, and showed that networks from some classic models do satisfy the small community phenomenon. A. Li and J. Li et al. [12] proposed a homophyly model by introducing a color for every vertex in the classical preferential attachment model such that networks generated from this model satisfy simultaneously the following properties: 1) power law degree distribution, 2) small diameter property, 3) vertices of the same color naturally form a small community, and 4) almost all vertices are contained in some small communities, i.e., the small community phenomenon of networks. This result implies the *homophyly law* of networks that the mechanism of the small community phenomenon is homophyly, and that vertices within a small community share remarkable common features.

* Corresponding author.
E-mail addresses: algzhang@sdu.edu.cn (P. Zhang), angsheng@ios.ac.cn (A. Li).

http://dx.doi.org/10.1016/j.tcs.2015.06.003
0304-3975/

A. Li and J. Li et al. [13] showed that many real networks satisfy exactly the homophyly law, in which an interesting application is the prediction and confirmation of keywords from a paper citation network of high energy physics theory.[1] The network contains 27,770 vertices (i.e., papers) and 352,807 edges (i.e., citations). All the papers have titles and abstracts, but only 1214 papers have keywords listed by their authors. We interpret the keywords of a paper to be a *function* of the paper. By the homophyly law, vertices within a small community of the network must share remarkable common features (keywords here). The prediction is as follows: 1) to find a small community from each vertex, if any, 2) to extract the most popular 5 keywords from the known keywords in a community, as the remarkable common features of this community, 3) to predict that (all or part of) the 5 remarkable common keywords are keywords of a paper in the community, 4) to confirm a prediction of keyword K for a paper P, if K appears in either the title or the abstract of paper P. It is a surprising result that this simple prediction confirms keywords for 19,200 papers in the network. This experiment implies that real networks do satisfy the homophyly law, and that the homophyly law is the principle for prediction in networks.

The keywords can be viewed as the attributes of vertices in a network. The above experimental result suggests a natural theoretical problem that, given a network in which some vertices have their attributes unfixed, how to assign attributes to these vertices such that the resulting network reflects the homophyly law in the most degree? Some attributes of a vertex cannot be changed, such as nationality, sex, color and language, but some other attributes can be changed, such as interest, job, income and working place. For simplicity, we consider the case that each vertex contains only one alterable attribute, i.e., the network is a 1-dimensional network. Consider the following scenario. Suppose in a company there are many employees which constitutes a friendship network. Some employees have been assigned to work in some departments of the company, while the remaining employees are waiting to be assigned. An employee is *happy*, if s/he works in the same department with all of (or ρ fraction of for some $\rho \in (0, 1]$, or at least q for some integer $q > 0$) her/his friends; otherwise s/he is *unhappy*. Similarly, a friendship is *happy* (or lucky) if the two related friends work in the same department; otherwise the friendship is *unhappy*. Our goal is to achieve the greatest social benefits, that is, to maximize the number of *happy vertices* (similarly, *happy edges*) in the network.

We can easily express the above problems as graph coloring problems, just identifying each attribute value with a different color. Hence we get two specific graph coloring problems, as defined below.

Definition 1 *(The MHV problem).* (Instance) In the Maximum Happy Vertices (MHV) problem, we are given an undirected graph $G = (V, E)$, a color set $C = \{1, 2, \cdots, k\}$, and a partial vertex coloring function $c: V \to C$. We say that c is a partial function in the sense that c assigns colors to part of vertices in V.

(Question) A vertex is *happy* if it shares the same color with all its neighbors, otherwise it is *unhappy*. The task is to extend c to a total function c' such that the number of happy vertices is maximized.

Definition 2 *(The MHE problem).* (Instance) The input of the Maximum Happy Edges (MHE) problem is the same as that of the MHV problem.

(Question) An edge is *happy* if its two endpoints have the same color, otherwise it is *unhappy*. The goal is to extend c to a total function c' such that the number of happy edges is maximized.

The vertex coloring defined by the total function $c': V \to C$ in MHV and MHE is called a *total vertex coloring*. In general, a (partial or total) vertex coloring can be denoted by $(V_1, V_2, \cdots, V_k)$, where V_i is the set of all vertices having color i. A total vertex coloring is a partition of $V(G)$, while a partial vertex coloring may not. Therefore, the MHV and MHE problems are two extension problems from a partial vertex coloring to a total vertex coloring. We remark that the coloring for our case is completely different from the well-known Graph Coloring problem, which requires that the two endpoints of an edge must be colored differently and asks to color a graph in such a way by using the minimized number of colors. We use the notion of color just for intuition.

Definition 3 *(k-MHV and k-MHE).* If in the MHV problem the color number k is a constant, the problem is denoted by k-MHV. Similarly, we have the k-MHE problem for constant k.

For the specific values of k, we have the 2-MHV problem, the 3-MHV problem, and so on. Similarly, we have the 2-MHE problem, the 3-MHE problem, etc that are the specific problems of k-MHE. Note that in the original MHV and MHE problems k is given as a part of the input.

We remark that both the MHV and MHE problems are natural and fundamental algorithmic problems, and that they have not appeared yet in literature. The reasons could be two folds. On the one hand, we ask the questions from our network applications which did not happen before; on the other hand, the meaning of coloring has been specified previously so that the two endpoints of an edge must have different colors. We notice that the current version of our problems may not really help network applications much because of their simplicity. For real network applications, probably the experimental method [13] introduced at the beginning of this section is fine enough. However, this has no theoretical guarantee, owing

[1] http://snap.stanford.edu/data/cit-HepTh.html.

to different structures of networks. Our problems seem essentially new and fundamental algorithmic problems. Theoretical analysis of the problems are always helpful to understand the nature of the problems, and hence are very welcome.

1.1. Our results

We investigate algorithms to solve the MHV and MHE problems. It is easy to see that the partial function c plays an important role in the MHV and MHE problems. If none of the vertices in the input graph has a pre-specified color, then the MHV and MHE problems are trivial. The optimal solution just assigns one arbitrary color to all the vertices. This will make all vertices and all edges happy.

We prove that the MHV and MHE problems are NP-hard. Interestingly, the complexity of k-MHV and k-MHE dramatically changes when k changes from 2 to 3. Specifically, we prove that both 2-MHV and 2-MHE can be solved in polynomial time, while both k-MHV and k-MHE are actually NP-hard for any constant $k \geq 3$. We thus seek approximation algorithms for the MHV and MHE problems, and their variants k-MHV and k-MHE ($k \geq 3$).

We design two approximation algorithms Greedy-MHV (Section 2.2.1) and Growth-MHV (Section 2.2.2) for the MHV problem and its variant k-MHV. Algorithm Greedy-MHV is a simple greedy algorithm with approximation ratio $1/k$. Algorithm Growth-MHV is an algorithm based on the subset-growth technique with approximation ratio $\Omega(\Delta^{-3})$, where Δ is the maximum degree of vertices in the input graph. In real networks, Δ is usually $\text{poly}\log n$, implying that the ratio $\Omega(\Delta^{-3})$ is reasonable. As Algorithm Growth-MHV is executing, more and more vertices are colored. According to the current vertex coloring for the input graph, we define several types for the vertices. (Note that the types here are not colors.) Algorithm Growth-MHV works based on carefully classifying all the vertices into several types.

We can extend our algorithms for MHV to deal with two more natural variants SoftMHV and HardMHV. In the SoftMHV problem, a vertex v is happy if v shares the same color with at least $\rho \deg(v)$ neighbors, where ρ (that is, the soft threshold) is a number in $(0, 1]$ and $\deg(v)$ is the degree of vertex v. In the HardMHV problem, a vertex v is happy if v shares the same color with at least q neighbors, where q (that is, the hard threshold) is an integer. We show that the SoftMHV and HardMHV problems can also be approximated within $\max\{1/k, \Omega(\Delta^{-3})\}$ by algorithms similar to that for MHV.

For the MHE problem and its variant k-MHE, we devise a simple approximation algorithm based on a division strategy, namely, Algorithm Division-MHE (Section 3). The approximation ratio is proved to be $1/2$.

1.2. Related work and relation to other problems

The MHV and MHE problems are two quiet natural vertex classification problems arising from the homophyly phenomenon in networks. Classification is a fundamental problem and has wide applications in statistics, pattern recognition, machine learning, and many other fields. Given a set of objects to be classified and a set of colors, a classification problem can be depicted as from a very high level assigning a color to each object in a way that is consistent with some observed data or structure that we have about the problem [1,11]. In our problems, the observed structure is homophyly. Since the MHV and MHE problems are essentially new, in the following we just show some closely related problems and results.

Thomas Schelling [17,18], the Nobel economics prize winner, showed by experiments how global patterns of spatial segregation arise from the effect of homophyly operating at the local level. The experiments in [17] are given in one-dimensional and two-dimensional geometric models. From a more general viewpoint of graph theory, Schelling's experiments, although given in geometric models, can be viewed as how to remove and add edges from/to a graph whose vertices are all colored by some colors such that the resulting graph possesses the homophyly property. In contrast, the MHV and MHE problems are how to color the vertices in a given graph whose part of vertices are already colored such that the resulting graph possesses the homophyly property.

The Multiway Cut problem [5,3,2,10] should be the traditional optimization problem that is most related to MHV and MHE. Given an undirected graph $G = (V, E)$ with costs defined on edges and a terminal set $S \subseteq V$, the Multiway Cut problem asks for a set of edges (called a *multiway cut*, or simply a *cut*) with the minimum total cost such that its removal from graph G separates all terminals in S from one another. The Multiway Cut problem in general graphs is NP-hard even the terminal set contains only three terminals and each edge has a unit cost [3]. The current best approximation ratio known for this problem is 1.3438 [10].

Removing a minimum multiway cut from a graph breaks the graph into several components such that each component contains exactly one terminal. From the viewpoint of graph coloring, this is equivalent to coloring the uncolored vertices in a graph in which each terminal has a distinct pre-specified color, such that the number of happy edges is maximized. Therefore, the MHE problem is actually the dual of the Multiway Cut problem. See Fig. 1 for an example. (More precisely, the dual of Multiway Cut is only a special case of MHE, since in MHE there may be more than one vertices having the same pre-specified color.) However, Multiway Cut and MHE are quite different in terms of approximation, since one is a maximization problem while the other is a minimization problem.

For a vertex subset $V' \subseteq V(G)$ of graph G, we define the *border* of V' to be the set of vertices in V' that has a neighbor not in V'. Given a vertex coloring $(V_1, V_2, \cdots, V_k)$ of graph G, the vertices in the border of each V_i are obviously unhappy. The MHV problem, which finds a vertex coloring that maximizes the number of happy vertices, is actually equivalent to finding a vertex coloring $(V_1, V_2, \cdots, V_k)$ for a graph in which some vertices are already colored, such that the total number

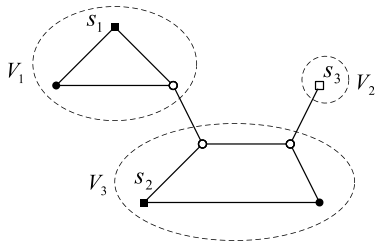

Fig. 1. An instance of Multiway Cut and the induced vertex coloring. The square vertices are terminals and have pre-specified colors, while the round vertices are non-terminal vertices. The hollow vertices are border vertices.

of vertices in borders of all V_i's is minimized. Please refer to Fig. 1 for an example. The latter problem we just introduce is a new minimization problem; the MHV problem and this new problem are dual to each other.

From the above analysis, one can see that the partial function c in the MHE problem (and the MHV problem), which assigns colors to part of vertices of the input graph, actually simulates and generalizes the *terminal set* part in the Multiway Cut problem.

Kann and Khanna et al. [9] studied the Max k-Cut problem [7] and its dual, that is, the Min k-Partition problem [9]. Given an undirected graph $G = (V, E)$, the Min k-Partition problem asks to find a vertex coloring $c: V \to \{1, 2, \cdots, k\}$ such that the number of edges whose two endpoints have the same color (i.e., the happy edges in our setting) is minimized.

According to the way of definitions in [9], we can define the dual of the Min k-Cut problem [16] as follows: Given an undirected graph $G = (V, E)$ and an integer $k > 0$, finding an edge subset whose removal breaks graph G into *exactly* k components, such that the number of remaining edges is maximized. Let's call this problem the Max k-Partition problem. In other words, Max k-Partition asks for a total vertex coloring $c': V \to \{1, 2, \cdots, k\}$ such that the number of happy edges is maximized, where c' should be a surjective function (that is, for each color i there exists a vertex whose color is i).

The Max k-Partition problem defined as above is close to the MHE problem, but they are still different in the obvious way: In Max k-Partition there is no any vertex having a pre-specified color and the required vertex coloring function c' must be surjective, while in MHE there must be some vertices having the pre-specified colors and the required vertex coloring function c' may not be surjective.

Notations. Let $G = (V, E)$ be a graph. Let $n = |V|$ and $m = |E|$. Suppose $v \in V$ is a vertex. Denote by $N(v)$ the set of neighbors of v. As usual, $\deg(v)$ means the degree of v, i.e., $\deg(v) = |N(v)|$. Denote by $N^2(v)$ the set of neighbors of neighbors of v (not including v itself), i.e., the vertices within distance 2 of v (assume each edge has unit distance).

Given a vertex coloring c, for a (colored or uncolored) vertex v, define $N^u(v)$ as the set of vertices in $N(v)$ that has not yet been colored. For a colored vertex v, define $N^s(v)$ as the set of vertices in $N(v)$ having the *same* color as $c(v)$, $N^d(v)$ as the set of vertices in $N(v)$ having colors *different* to $c(v)$.

Given an instance $\mathcal{I}$ of some optimization problem $\mathcal{P}$, we use $OPT(\mathcal{I})$ (OPT for short) to denote the optimum (that is, the value of an optimal solution) of the instance. Let $\mathcal{A}$ be an algorithm for problem $\mathcal{P}$. We use $SOL(\mathcal{I})$ (SOL for short) to denote the value of the solution found by algorithm $\mathcal{A}$ on instance $\mathcal{I}$ of problem $\mathcal{P}$. In addition, OPT and SOL also denote the corresponding solutions, abusing notations slightly.

Organization of the paper. The remaining of the paper is organized as follows. In Section 2, we show that 2-MHV is polynomial-time solvable, and give the greedy approximation algorithm and the subset-growth approximation algorithm for the MHV and k-MHV ($k \geq 3$) problems. In Section 3, we show that 2-MHE is polynomial-time solvable, and give the division-strategy based approximation algorithm for the MHE and k-MHE ($k \geq 3$) problems. In Section 4, we prove the NP-hardness for the MHE, k-MHE ($k \geq 3$), MHV, and k-MHV ($k \geq 3$) problems. In Section 5, we give approximation results for the SoftMHV and HardMHV problems, while their proofs are deferred to Appendix A. Finally, we conclude the paper in Section 6 by introducing some future work.

2. Algorithms for MHV

In Section 2.1, we give the polynomial time exact algorithm for the 2-MHV problem. In Section 2.2, we give the approximation algorithms Greedy-MHV and Growth-MHV for the MHV problem.

2.1. 2-MHV is in P

Let U be a finite set. Recall that a function $f: 2^U \to \mathbf{Z}^+$ is said to be submodular if $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ holds for all $X, Y \subseteq U$. Given a vertex subset $V' \subseteq V(G)$, define function $f(V')$ to be the number of vertices in V' that has neighbors outside of V', i.e., $f(V')$ is the size of the border (see Section 1.2) of V'. It is easy to verify that f is a submodular function.

Consider the 2-MHV problem, in which the color set C contains only two colors 1 and 2. This problem can be solved in polynomial time.

Theorem 1. *The 2-MHV problem can be solved in $O(mn^7 \log n)$ time.*

Proof. Let V_1^{org} be the set of vertices that are colored by color 1 by the partial function c, and V_2^{org} be the analogous vertex subset corresponding to color 2. Then the 2-MHV problem is equivalent to finding a cut (V_1, V_2) such that $V_i^{org} \subseteq V_i$ for $i = 1, 2$ and $f(V_1) + f(V_2)$ is minimized. We can do this by merging all vertices in V_1^{org} to a single vertex s, all vertices in V_2^{org} to a single vertex t, and finding an s–t cut (V_1, V_2) on the resulting graph such that $f(V_1) + f(V_2)$ is minimized. As pointed out by [19, Lemma 3], finding such a cut can be done by an algorithm in [8] for minimizing submodular functions in $O(\theta n^7 \log n)$ time, where θ is the time to compute the submodular function f. When the input graph is stored by a collection of adjacency lists, $f(\cdot)$ can be computed in $O(m)$ time in a straightforward way (assuming the input graph contains no isolated vertex). The proof of the theorem is finished. □

2.2. Approximation algorithms for MHV

The approximation algorithms for MHV work based on the types defined for vertices, as shown in Definition 4.

Definition 4 *(Types of vertices in MHV).* Fix a (partial or total) vertex coloring. Let v be a vertex. Then,

1. v is an *H-vertex* if v is colored and happy (i.e., $|N^s(v)| = \deg(v)$);
2. v is a *U-vertex* if v is colored and destined to be unhappy (i.e., $|N^d(v)| > 0$);
3. v is a *P-vertex* if
   (a) v is colored,
   (b) v has not been happy (i.e., $|N^s(v)| < \deg(v)$), and
   (c) v may become happy in the future (i.e., $|N^d(v)| = 0$);
4. v is an *L-vertex* if v has not been colored.

See Figs. 2, 3, 4 for examples of the vertex types. Note that by a type name we also mean the set of vertices of that type. Conversely, by a set name we also mean that each element in the set is of that type. For example, H is the set of all H-vertices; each vertex in the set H is an H-vertex.

2.2.1. Greedy approximation algorithm for MHV

Algorithm *(*Greedy-MHV*).* The approximation algorithm Greedy-MHV for MHV is quiet simple. We just color all uncolored vertices by the same color. Since there are k colors in C, we can obtain k vertex colorings for graph G. Finally we output the coloring that has the most number of happy vertices.

Theorem 2. *Algorithm* Greedy-MHV *is a $1/k$-approximation algorithm for the MHV problem, where k is the number of colors given in the input.*

Proof. Let the partial function c be the vertex coloring used in Definition 4. We partition L-vertices further into two subsets L_P and L_U. L_P is the set of uncolored vertices that can become happy (i.e., whose neighbors have at most one color). L_U is the set of uncolored vertices that are destined to be unhappy (i.e., whose neighbors already have at least two distinct colors). Then (H, P, U, L_P, L_U) is a partition of $V(G)$. Obviously, in the best case *OPT* can make all vertices in S, P and L_P happy, implying $|H| + |P| + |L_P| \geq OPT$.

Let SOL_i be the number of happy vertices when Algorithm Greedy-MHV colors all uncolored vertices by color i. Then we have $|H| + |P| + |L_P| \leq \sum_i SOL_i$. By the greedy strategy, *SOL*, which is the number of happy vertices found by Greedy-MHV, is at least $\frac{1}{k}(|H| + |P| + |L_P|)$. The theorem follows by observing that Greedy-MHV obviously runs in polynomial time. □

2.2.2. Subset-growth approximation algorithm for MHV

The subset-growth algorithm starts with the partial vertex coloring $(V_1, V_2, \cdots, V_k)$ defined by the partial function c. From a high level point of view, the algorithm iteratively augments the subsets in $(V_1, V_2, \cdots, V_k)$ by satisfying the vertices that can become happy easily at the current time, until $(V_1, V_2, \cdots, V_k)$ becomes a partition of $V(G)$ and thus a vertex coloring is obtained. This strategy is based on the following further classification of L-vertices, according to the type of their neighbors. Recall that by Definition 4, L-vertex means uncolored vertex.

Definition 5 *(Subtypes of L-vertex in MHV).* Let v be an L-vertex in a vertex coloring. Then,

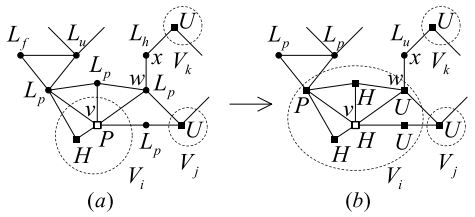

Fig. 2. Process a P-vertex. The hollow vertex v in graph (a) is the P-vertex to be processed. The square vertices mean colored vertices, while the round vertices mean uncolored vertices.

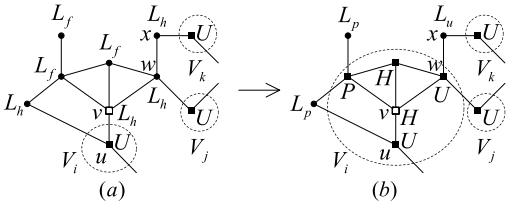

Fig. 3. Process an L_h-vertex. The hollow vertex v in graph (a) is the L_h-vertex to be processed. Note that when an L_h-vertex is to be processed, there is no P-vertex in the current graph (a).

1. v is an L_p-*vertex* if v is adjacent to a P-vertex;
2. v is an L_h-*vertex* if
   (a) v is not adjacent to any P-vertex,
   (b) v can become happy, that is, v is adjacent to U-vertices with only one color;
3. v is an L_u-*vertex* if
   (a) v is not adjacent to any P-vertex,
   (b) v is destined to be unhappy, that is, v is adjacent to U-vertices with more than one colors;
4. v is an L_f-*vertex* if v is not adjacent to any colored vertex.

See Figs. 2, 3, 4 for examples of the subtypes of L-vertex.
The subset-growth algorithm GROWTH-MHV is as follows.

Algorithm GROWTH-MHV
Input: A connected undirected graph G and a partial coloring function c.
Output: A total vertex coloring for G.

1. $\forall 1 \le i \le k$, $V_i \leftarrow \{v : c(v) = i\}$.
2. **while** there exist L-vertices **do**
   (a) **if** there exists a P-vertex v **then**
      i. $i \leftarrow c(v)$.
      ii. Add all the L_p-neighbors of v to V_i. The types of all affected vertices (including v and vertices in $N^2(v)$) are changed accordingly.
   (b) **elseif** there exists an L_h-vertex v **then**
      i. Let u be any U-vertex adjacent to v, then $i \leftarrow c(u)$.
      ii. Add v and all its L-neighbors to V_i. The types of all affected vertices (including v and vertices in $N^2(v)$) are changed accordingly.
   (c) **else**
      Comment: There must be an L_u-vertex.
      i. Let v be any L_u-vertex, u be the any U-vertex adjacent to v, then $i \leftarrow c(u)$.
      ii. Add v to V_i. The types of all affected vertices (including v and vertices in $N(v)$) are changed accordingly.
   (d) **endif**
3. **endwhile**
4. **return** the vertex coloring $(V_1, V_2, \cdots, V_k)$.

When there are still L-vertices (i.e., uncolored vertices), Algorithm GROWTH-MHV works in the following way. It first colors a P-vertex's neighbors to make this P-vertex happy (see Fig. 2). When there is no any P-vertex, it colors an L_h-vertex and its neighbors to make the L_h-vertex happy (see Fig. 3). When there is no any P-vertex or L_h-vertex, it colors an L_u-vertex by the color of its any U-vertex neighbor (see Fig. 4). Note that coloring a vertex may generate new P-vertices, or L_h-vertices, or L_u-vertices.

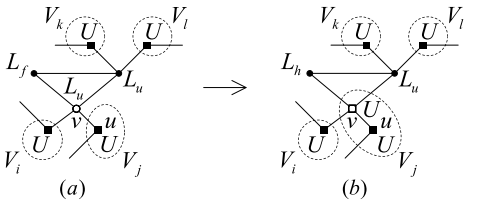

Fig. 4. Process an L_u-vertex. The hollow vertex v in graph (a) is the L_u-vertex to be processed. Note that when an L_u-vertex is to be processed, there is no any P-vertex or L_h-vertex in the current graph (a).

When there exist L-vertices, it is impossible that there are only L_f-vertices but no any L_p-vertex, L_h-vertex or L_u-vertex, since by assumption G is a connected graph and by definition L_f-vertex is not adjacent to any colored vertex. So, when there isn't any L_p-vertex or L_h-vertex, there must be at least one L_u-vertex. As a result, in step (2c) we don't need an **if** statement like that in steps (2a) and (2b).

We use a type name with the superscript "org" (means "original") to denote the set of vertices of that type which is determined by the partial function c, and a type name with the superscript "new" to denote the set of vertices of that type which is determined in the execution of Algorithm Growth-MHV. For example, H^{org} is the set of H-vertices that are determined by the partial function c, and H^{new} is the set of H-vertices that are newly generated by Algorithm Growth-MHV.

Let Δ be the maximum degree of vertices in the input graph. We first bound the number of L_u^{new}-vertices.

Lemma 3. $|L_u^{new}| \leq \Delta(\Delta-2)|H^{new}|$.

Proof. Algorithm Growth-MHV iteratively processes three types of vertices, that is, the P-vertices, the L_h-vertices and the L_u-vertices. We will prove the lemma by proving the following three points: (1) When Algorithm Growth-MHV processes a P-vertex, at most $\Delta(\Delta-2)$ L_u^{new}-vertices are generated, (2) When Algorithm Growth-MHV processes an L_h-vertex, at most $(\Delta-1)(\Delta-2)$ L_u^{new}-vertices are generated, and (3) When Algorithm Growth-MHV processes an L_u-vertex, no L_u^{new}-vertex is generated.

Consider the first point. Let v be a P-vertex to be processed. Suppose v has an L_p-neighbor w, which is adjacent to a U-vertex. Only if there is an L_h-vertex x which is the neighbor of w, x will become a newly generated L_u-vertex when the P-vertex v is processed. See Fig. 2 for an example. Since the maximum vertex degree is Δ, v has at most Δ L_p-neighbors, and w has at most $\Delta-2$ L_h-neighbors. This implies that when v is processed, at most $\Delta(\Delta-2)$ L_u^{new}-vertices can be generated.

Then consider the second point. Suppose the L_h-vertex to be processed is v. Suppose v has an L-neighbor w (w can be an L_h-vertex or an L_u-vertex), which is adjacent to a U-vertex. Similarly, only if there is an L_h-vertex x which is the neighbor of w, x will become a newly generated L_u-vertex when the L_h-vertex v is processed. See Fig. 3 for an example. Since the maximum vertex degree is Δ, v has at most $\Delta-1$ L-neighbors, and w has at most $\Delta-2$ L_h-neighbors. This implies that when v is processed, at most $(\Delta-1)(\Delta-2)$ L_u^{new}-vertices can be generated.

Finally consider the third point. When Algorithm Growth-MHV processes an L_u-vertex, there is no any L_h-vertex (or P-vertex) in the current graph. So, adding an L_u-vertex to some subset V_i does not generate any new L_u-vertex. See Fig. 4 for an example.

When Algorithm Growth-MHV processes a P-vertex or an L_h-vertex, at least one vertex becomes an H-vertex. So we can charge the number of newly generated L_u-vertices to this newly generated H-vertex. This finishes the proof of the lemma. □

The following Lemma 4 gives an upper bound on OPT, the number of happy vertices in an optimal solution to the k-MHV problem.

Lemma 4. $OPT \leq |H^{org}| + (\Delta+1)(|L^{org}| - |L_u^{org}|)$.

Proof. By the partial function c, all vertices in the original graph (i.e., the input graph that has not been colored by Algorithm Growth-MHV) are partitioned into four vertex subsets H^{org}, P^{org}, U^{org} and L^{org}. Subset L^{org} is further partitioned into four subsets L_p^{org}, L_h^{org}, L_u^{org} and L_f^{org}. By definition, all vertices in U^{org} are unhappy. And, all vertices in L_u^{org} are destined to be unhappy since each of them is adjacent to at least two vertices with different colors. So, in the best case all vertices in P^{org} and L^{org} except those in L_u^{org} would be happy. Noticing that the vertices in H^{org} are already happy, we have

$$OPT \leq |H^{org}| + |P^{org}| + |L^{org}| - |L_u^{org}|. \tag{1}$$

Since each P-vertex must be adjacent to some L_p-vertex, and each L_p-vertex can be adjacent to at most Δ P-vertices, the number of P^{org}-vertices is at most $\Delta|L_p^{org}|$. Since $|L_p^{org}| \leq |L^{org}| - |L_u^{org}|$, we get that

$$\begin{aligned} OPT &\le |H^{org}| + \Delta|L_p^{org}| + |L^{org}| - |L_u^{org}| \\ &\le |H^{org}| + (\Delta+1)(|L^{org}| - |L_u^{org}|), \end{aligned}$$

concluding the lemma. □

Lemma 5. $|H^{new}| \ge \frac{1}{\Delta(\Delta-1)}(|L^{org}| - |L_u^{org}|)$.

Proof. Recall that there are four subtypes of an L-vertex, i.e., L_p-vertex, L_h-vertex, L_u-vertex and L_f-vertex. Among them only L_p-vertex and L_h-vertex will (directly) contribute to generating H-vertices. For an L_f-vertex, it will ultimately become one of the other three types of L-vertex. For each L_u-vertex, although it may become an L_p-vertex and hence can contribute to generating H-vertices, in the worst case we may assume that it is added to some subset V_i and contribute nothing to the generation of H-vertex.

By step (2a) and step (2c), each time an H-vertex is generated, at most Δ L_p-vertices or L_h-vertices are consumed (i.e., colored). Furthermore, once an L-vertex is colored, it will never be re-colored or de-colored. So we have

$$|H^{new}| \ge \frac{1}{\Delta}(|L^{org}| - |L_u^{org}| - |L_u^{new}|).$$

By Lemma 3, we have

$$\begin{aligned} \frac{1}{\Delta}(|L^{org}| - |L_u^{org}| - |L_u^{new}|) &\ge \frac{1}{\Delta}(|L^{org}| - |L_u^{org}| - \Delta(\Delta-2)|H^{new}|) \\ &= \frac{1}{\Delta}(|L^{org}| - |L_u^{org}|) - (\Delta-2)|H^{new}|. \end{aligned}$$

Therefore, $(\Delta-1)|H^{new}| \ge \frac{1}{\Delta}(|L^{org}| - |L_u^{org}|)$. The lemma follows. □

Theorem 6. *The MHV problem can be approximated within a factor of $\Omega(\Delta^{-3})$ in polynomial time.*

Proof. Algorithm Growth-MHV obviously runs in polynomial time. Let *SOL* be the number of happy vertices found by Algorithm Growth-MHV. Then we have

$$\begin{aligned} SOL &= |H^{org}| + |H^{new}| \\ &\ge |H^{org}| + \frac{1}{\Delta(\Delta-1)}\Big(|L^{org}| - |L_u^{org}|\Big) \quad \text{(By Lemma 5)} \\ &\ge \frac{1}{\Delta(\Delta-1)(\Delta+1)}\Big(|H^{org}| + (\Delta+1)(|L^{org}| - |L_u^{org}|)\Big) \\ &\ge \frac{1}{\Delta(\Delta-1)(\Delta+1)} OPT \qquad \text{(By Lemma 4)} \\ &= \Omega(\Delta^{-3})OPT. \end{aligned}$$

The theorem follows. □

3. Algorithms for MHE

3.1. 2-MHE is in P

For 2-MHE, the partial function c can only use two colors, to say, color 1 and color 2. Given such an instance, merge all vertices with color 1 assigned by c into a single vertex s, and all vertices with color 2 into a single vertex t. (The edges whose two endpoints are merged disappear in the procedure.) Then compute a minimum s–t cut (V_1, V_2) on the resulting instance. Suppose $s \in V_1$ and $t \in V_2$. Assign color 1 to all vertices (including the merged vertices) in V_1, and color 2 to all vertices in V_2. Since (V_1, V_2) is a minimum s–t cut, the number of happy edges in the resulting vertex coloring is maximized. By the work of [6], a maximum flow (and hence a minimum s–t) in a unit capacity network can be computed in $O(\min\{n^{2/3}m, m^{3/2}\})$ time. So we have

Theorem 7. *The 2-MHE problem can be solved in $O(\min\{n^{2/3}m, m^{3/2}\})$ time.* □

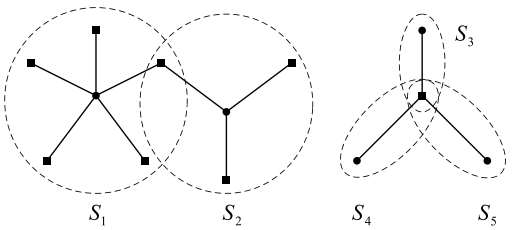

Fig. 5. An example of graph G'. Each edge in G' has its one endpoint colored and the other endpoint uncolored. The square vertices mean colored vertices, while the round vertices mean uncolored vertices. Each star (marked with dashed circle) is centered at an uncolored vertex. Two stars (e.g., S_1 and S_2) may share common colored vertices.

3.2. Approximation algorithm for MHE

The MHE problem admits a simple division-strategy based algorithm which yields a 1/2-approximation. The algorithm is designed to deal with more general graphs with nonnegative weights $\{w(e)\}$ defined on edges. We thus denote by $w(E')$ the total weight of edges in an edge subset E'.

Algorithm DIVISION-MHE
Input: An undirected graph G and a partial coloring function c.
Output: A total vertex coloring for G.

1. $G_1 \leftarrow G$.
2. Let E' be the set of edges in G_1 that has exactly one endpoint not colored by function c. Define graph $G' = (V(G_1), E')$, which is a subgraph of G_1.
3. For each star S in G' centered at an uncolored vertex v, color v by a color in $\{c(u) \mid u \in N(v), u \text{ is colored}\}$ such that the total weight of happy edges in S is maximized.
4. Color all vertices in G_1 still having not been colored by just one arbitrary color. Denote by SOL_1 the vertex coloring of G_1.
5. $G_2 \leftarrow G$.
6. Color all uncolored vertices in G_2 by just one arbitrary color. Denote by SOL_2 the vertex coloring of G_2.
7. **return** the better one among SOL_1 and SOL_2.

Algorithm DIVISION-MHE computes two independent solutions SOL_1 and SOL_2 to graph G, and then outputs the better one, where the better one means the solution making more edges happy. For an illustration of graph G' and its stars in step (3), please refer to Fig. 5.

Theorem 8. *Algorithm* DIVISION-MHE *is a* 1/2*-approximation algorithm for the MHE problem.*

Proof. First, the algorithm obviously runs in polynomial time.

Let W^{org} be the total weight of edges already being happy by the partial coloring function c. This weight can be trivially obtained by any solution.

Let W' be the total weight of happy edges found by Algorithm DIVISION-MHE on graph G'. Note that W' is the maximum total weight that can be obtained from graph G'. Let E'' be the set of edges that has both of its two endpoints uncolored by function c, and $W'' = w(E'')$ be its total weight. Then we have $OPT \leq W^{org} + W' + W''$.

By the algorithm, we know $SOL_1 \geq W^{org} + W'$ and $SOL_2 \geq W^{org} + W''$. Then the approximation ratio 1/2 of DIVISION-MHE is obvious since $SOL = \max\{SOL_1, SOL_2\} \geq \frac{1}{2}(W^{org} + W' + W'')$. □

4. Hardness results

4.1. NP-hardness of MHE

The NP-hardness of the 3-MHE problem is proved by a reduction from the Multiway Cut problem [3].

Theorem 9. *The 3-MHE problem is NP-hard.*

Proof. Given an undirected graph $G = (V, E)$ and a terminal set $D = \{s_1, s_2, s_3\}$, the 3-Terminal Cut problem (i.e., the Multiway Cut problem with 3 terminals), which is NP-hard [3], asks for a minimum cardinality edge set such that its removal from G disconnects the three terminals from one another. Given an instance (G, D) of 3-Terminal Cut, we construct the instance (H, C, c) of 3-MHE as follows. Graph H is just G. Color set C is set to be $\{1, 2, 3\}$. The partial function c assigns colors 1, 2, 3 to vertices s_1, s_2, s_3, respectively. Let c^* be the cardinality of an optimal 3-way cut for (G, D), and m^* be

the number of happy edges of an optimal vertex coloring for (H, C, c). Then one can easily find that $m^* = m - c^*$, where $m = |E(G)|$ $(= |E(H)|)$. This shows the 3-MHE problem is NP-hard. □

Corollary 1. *The MHE problem is NP-hard.*

Proof. In the input of MHE, just set k to be 3. □

Theorem 10. *The k-MHE problem is NP-hard for any constant $k \geq 3$.*

Proof. By Theorem 9, we need only focus on $k > 3$. Let k be such a constant.

Given a 3-MHE instance (G, c), we construct a k-MHE instance (G', c') as follows. Build $2(k-3)$ vertices $x_4, y_4, x_5, y_5, \cdots, x_k, y_k$ and $k-3$ edges (x_i, y_i), $4 \leq i \leq k$. Vertices x_i and y_i are colored by color i, for $4 \leq i \leq k$. Let v be a vertex in G whose color given by c is 1. Then put $k-3$ edges (v, x_i), $4 \leq i \leq k$. This is our new instance (G', c').

Obviously for $4 \leq i \leq k$, each edge (x_i, y_i) is happy whereas each edge (v, x_i) is unhappy. So, the optimum of (G, c) is just equal to the optimum of (G', c') minus $k-3$, concluding the theorem. □

4.2. NP-hardness of MHV

Theorem 11. *The k-MHV problem is NP-hard for any constant $k \geq 3$.*

Proof. By Theorem 10, k-MHE is NP-hard ($k \geq 3$). We thus reduce k-MHE to k-MHV.

Let (G, c) be a k-MHE instance. The instance (G', c') of k-MHV is constructed as follows. Add k vertices $x_1, x_2, \cdots, x_k$ and put an edge between x_i and v, for each $1 \leq i \leq k$ and each $v \in V(G)$. Vertex x_i is colored by i, for $1 \leq i \leq k$. For every edge $(u, v) \in E(G)$, add a vertex y_{uv} and replace the edge by two edges (u, y_{uv}) and (y_{uv}, v). This is our new instance (G', c').

Since in graph G there are vertices with pre-specified colors, each x_i $(1 \leq i \leq k)$ cannot become happy no matter how the remaining vertices are colored. Every original vertex $v \in V(G)$ also cannot become happy since it is adjacent to all x_i's. Let (u, v) be any edge in G. Since the degree of vertex y_{uv} is 2, it is happy iff its two neighbors have the same color. This shows that the optimum of the k-MHE instance (G, c) is equal to the optimum of the k-MHV instance (G', c'). The theorem follows. □

5. Extensions and variants of MHV

There exist natural extensions and variants for the MHV problem. For a vertex v in the MHV problem, instead of requiring that *all* neighbors of v have the same color as that of v, to make v happy we may only require at least $\rho \cdot \deg(v)$ neighbors have the same color as that of v, or only require at least q neighbors have the color identical to that of v, for some global number q. This leads to two natural variants of the MHV problem, that is, the SoftMHV problem and the HardMHV problem.

Let ρ be an input number in $(0, 1]$. In the soft-threshold extension of the MHV problem (SoftMHV for short), a vertex v is happy if v is colored and $|N^s(v)| \geq \rho \cdot \deg(v)$.

Definition 6 *(The SoftMHV problem).* (Instance) Given a connected undirected graph $G = (V, E)$, a color set $C = \{1, 2, \cdots, k\}$, a partial coloring function $c: V \to C$, and a number $\rho \in (0, 1]$, (Question) the SoftMHV problem asks for a total vertex coloring extended from c that maximizes the number of happy vertices.

It is easy to see that MHV is a special case of SoftMHV in which $\rho = 1$.

In the hard-threshold variant of the MHV problem (HardMHV for short), a vertex v is happy if $|N^s(v)| \geq q$, where $q \in \mathbb{Z}^+$ is an input number.

Definition 7 *(The HardMHV problem).* (Instance) Given a connected undirected graph $G = (V, E)$, a color set $C = \{1, 2, \cdots, k\}$, a partial coloring function $c: V \to C$, and an integer $q > 0$, (Question) the HardMHV problem asks for a total vertex coloring extended from c that maximizes the number of happy vertices.

It is reasonable to assume $q \leq \Delta$ in HardMHV, since otherwise there is no feasible solution to the problem. Also note that MHV is *not* a special case of HardMHV since in HardMHV q is a global parameter that has nothing to do with the degree of vertices.

We can extend Algorithm Greedy-MHV and Algorithm Growth-MHV to deal with the SoftMHV and HardMHV problems, and thus have

Theorem 12. *Both SoftMHV and HardMHV can be approximated within* $\max\{1/k, \Omega(\Delta^{-3})\}$ *in polynomial time.* □

The approximation algorithms used in Theorem 12 are similar to that for the MHV problem. For completeness, they are given in Appendix A (See Theorems 13, 15, 16, and 18).

Similarly, we can define the corresponding soft-threshold extension and hard-threshold variant for the k-MHV problem, and our results in this section naturally extend to them. We just omit the details for simplicity.

6. Conclusions

The MHV problem and the MHE problem are two natural graph coloring problems arising in the homophyly phenomenon of networks. In this paper we prove the NP-hardness of the MHV problem and the MHE problem, and give several approximation algorithms for these two problems.

Since our algorithms Greedy-MHV, Growth-MHV and Division-MHE actually do not care whether the color number k is given in the input or whether k is a constant, the k-MHV and k-MHE problems can also be approximated within $\max\{1/k, \Omega(\Delta^{-3})\}$ and $1/2$, respectively.

To improve the approximation ratios for MHV and MHE remains an immediate open problem. It is also interesting to study the MHV and MHE problems in random graphs generated from the classical network models, and in the real-world large networks.

Acknowledgements

We are grateful for fruitful discussions on this paper with Dr. Mingji Xia at Institute of Software, Chinese Academy of Sciences.

Angsheng Li is supported by the hundred talent program of the Chinese Academy of Sciences, and the grand challenge program, *Network Algorithms and Digital Information*, Institute of Software, Chinese Academy of Sciences. Peng Zhang is supported by the National Natural Science Foundation of China (60970003), and the Independent Innovation Foundation of Shandong University (2012TS072).

Appendix A

Fix a vertex coloring, and let v be a (colored or uncolored) vertex. Define $N_i(v)$ to be the set of vertices in $N(v)$ which has color i, for $1 \leq i \leq k$.

A.1. Approximation algorithms for SoftMHV

As what is done in Definition 4, we define the types of vertices according to the given vertex coloring.

Definition 8 *(Types of vertex in SoftMHV).* Fix a (partial or total) vertex coloring. Let v be a vertex. Then,

1. v is an H-vertex if v is colored and happy;
2. v is a U-vertex if
   (a) v is colored, and
   (b) v is destined to be unhappy, (i.e., $\deg(v) - |N^d(v)| < \rho \cdot \deg(v)$);
3. v is a P-vertex if
   (a) v is colored,
   (b) v has not been happy (i.e., $|N^s(v)| < \rho \cdot \deg(v)$), and
   (c) v can become an H-vertex (i.e., $|N^s(v)| + |N^u(v)| \geq \rho \cdot \deg(v)$);
4. v is an L-vertex if v has not been colored.

We note that Algorithm Greedy-MHV is also a $1/k$-approximation algorithm for the SoftMHV problem. To see this, we just define L_P in Theorem 2 as the set of uncolored vertices v such that $|N^u(v)| + \max\{|N_i(v)|\} \geq \rho \cdot \deg(v)$, and $L_D = L - L_P$.

Theorem 13. *The SoftMHV problem can be approximated within a factor of* $1/k$ *in polynomial time.* □

Below we give the subset-growth approximation algorithm Growth-SoftMHV for the SoftMHV problem. First we define the subtypes of L-vertex.

Definition 9 *(Subtypes of L-vertex in SoftMHV).* Let vertex v be an L-vertex in a vertex coloring. Then,

1. v is an L_p-vertex if v is adjacent to a P-vertex,
2. v is an L_h-vertex if
   (a) v is not adjacent to any P-vertex,

(b) v is adjacent to an H-vertex or a U-vertex, and
(c) v can become happy (that is, $|N^u(v)| + \max\{|N_i(v)|: 1 \le i \le k\} \ge \rho \cdot \deg(v)$),
3. v is an L_u-vertex if
(a) v is not adjacent to any P-vertex,
(b) v is adjacent to an H-vertex or a U-vertex, and
(c) v is destined to be unhappy (that is, $|N^u(v)| + \max\{|N_i(v)|: 1 \le i \le k\} < \rho \cdot \deg(v)$),
4. v is an L_f-vertex if v is not adjacent to any colored vertex.

Algorithm GROWTH-SOFTMHV
Input: A connected undirected graph G and a partial coloring function c.
Output: A total vertex coloring for G.

1. $\forall 1 \le i \le k$, $V_i \leftarrow \{v: c(v) = i\}$.
2. **while** there exist L-vertices **do**
(a) **if** there exists a P-vertex v **then**
i. $i \leftarrow c(v)$.
ii. Add its any $\lceil \rho \cdot \deg(v) \rceil - |N^s(v) \cap V_i|$ L_p-neighbors to vertex subset V_i. The types of all affected vertices (including v and vertices in $N^2(v)$) are changed accordingly.
(b) **elseif** there exists an L_h-vertex v **then**
i. Let V_i be the vertex subset in which v has the maximum colored neighbors.
ii. Add vertex v and its any $\lceil \rho \cdot \deg(v) \rceil - |N^s(v) \cap V_i|$ L-neighbors to vertex subset V_i. The types of all affected vertices (including v and vertices in $N^2(v)$) are changed accordingly.
(c) **else**
Comment: There must be an L_u-vertex.
i. Let v be any L_u-vertex, and V_i be any vertex subset in which v has colored neighbors.
ii. Add vertex v to subset V_i. The types of all affected vertices (including v and vertices in $N(v)$) are changed accordingly.
(d) **endif**
3. **endwhile**
4. **return** the vertex coloring $(V_1, V_2, \cdots, V_k)$.

In step (2(a)ii), the algorithm adds the least number (that is, $\lceil \rho \cdot \deg(v) \rceil - |N^s(v) \cap V_i|$) of v's neighbors to subset V_i to make v happy. The same thing is done in step (2(b)ii).

Lemma 14. $|L_u^{new}| \le O(\Delta^2)|H^{new}|$.

Proof. Suppose Algorithm GROWTH-SOFTMHV is to process a P-vertex v, which is already colored by color i. When v is processed, at most $\lceil \rho\Delta \rceil$ L_p-neighbors of v are added to V_i. Each of the L_p-neighbors has at most $\Delta - 1$ L_h-neighbors. In the worst case, all these L_h-neighbors, plus the remaining L_p-neighbors of v, could become L_u-vertices when v is processed. So, at most $\lceil \rho\Delta \rceil(\Delta - 1) + (1 - \alpha)\Delta = O(\Delta^2)$ L_u^{new}-vertices can be generated in this case.

Then suppose the algorithm is to process an L_h-vertex v. Let V_i be the vertex subset in which v has the maximum colored neighbors. When v is processed, at most $\lceil \rho\Delta \rceil - 1$ L-neighbors of v are added to V_i. Each of these L-neighbors can have at most $\Delta - 1$ L_h-neighbors. In the worst case, all these L_h-neighbors, plus the remaining L-neighbors of v, could become L_u-vertices when v is processed. So, at most $(\lceil \rho\Delta \rceil - 1)(\Delta - 1) + (1 - \alpha)\Delta = O(\Delta^2)$ L_u^{new}-vertices can be generated in this case.

When the algorithm processes an L_u-vertex, there are only L_u-vertices or L_f-vertices (if any) in the current graph. So, coloring an L_u-vertex does not generate any new L_u-vertex.

By charging the number of newly generated L_u-vertices to the newly generated H-vertex, we finish the proof of the lemma. □

Theorem 15. *The SoftMHV problem can be approximated within a factor of* $\Omega(\Delta^{-3})$ *in polynomial time.*

Proof. Each time an H-vertex is generated, at most $\lceil \rho\Delta \rceil$ L-vertices are consumed (i.e., colored). So, for the number of newly generated H-vertices we have $|H^{new}| \ge (|L^{org}| - |L_u^{org}| - |L_u^{new}|)/\lceil \rho\Delta \rceil$. By Lemma 14, we get

$$|H^{new}| \ge \frac{|L^{org}| - |L_u^{org}|}{O(\Delta^2)}. \tag{A.1.1}$$

Let *OPT* be the number of happy vertices in an optimal solution to the problem. By the same reason as in Lemma 4, we obtain

$$\begin{aligned} OPT &\le |H^{org}| + |P^{org}| + |L^{org}| - |L_u^{org}| \\ &\le |H^{org}| + \Delta|L_p^{org}| + |L^{org}| - |L_u^{org}| \\ &\le |H^{org}| + (\Delta+1)(|L^{org}| - |L_u^{org}|). \end{aligned}$$

Let *SOL* be the number of happy vertices found by Algorithm Growth-SoftMHV. Then we have

$$\begin{aligned} SOL &= |H^{org}| + |H^{new}| \\ &\ge |H^{org}| + \frac{1}{O(\Delta^2)}\Big(|L^{org}| - |L_u^{org}|\Big) \\ &\ge \frac{1}{O(\Delta^3)}\Big(|H^{org}| + \Delta(|L^{org}| - |L_u^{org}|)\Big) \\ &= \Omega(\Delta^{-3})OPT. \end{aligned}$$

Finally, notice that Algorithm Growth-SoftMHV obviously runs in polynomial time. This gives the theorem. □

A.2. Approximation algorithms for HardMHV

The following type definition of vertices is similar to Definition 8.

Definition 10 *(Types of vertex in HardMHV).* Fix a (partial or total) vertex coloring. Let v be a vertex. Then,

1. v is an H-vertex if v is colored and happy,
2. v is a U-vertex if
   (a) v is colored, and
   (b) v is destined to be unhappy (i.e., $\deg(v) - |N^d(v)| < q$),
3. v is a P-vertex if
   (a) v is colored,
   (b) v has not been happy (that is, $|N^s(v)| < q$), and
   (c) v can become happy (i.e., $|N^s(v)| + |N^u(v)| \ge q$),
4. v is an L-vertex if v has not been colored.

Similar as the case of SoftMHV, Algorithm Greedy-MHV is also a $1/k$-approximation algorithm for the HardMHV problem. To prove this we only need to define L_P in Theorem 2 as the set of uncolored vertices v such that $|N^u(v)| + \max\{|N_i(v)|\} \ge q$, and $L_D = L - L_P$.

Theorem 16. *There is a $1/k$-approximation algorithm for the HardMHV problem.* □

In the MHV and SoftMHV problems, for an L-vertex v, if $|N^d(v)|$ is too large, then v may be destined to be unhappy. In contrast, in the HardMHV problem, an L-vertex v may be destined to be unhappy even if $|N^d(v)| = 0$: This will happen when $\deg(v) < q$. Based on this observation, the L-vertex type is divided into the following four subtypes.

Definition 11 *(Subtypes of L-vertex in HardMHV).* Let vertex v be an L-vertex in a vertex coloring. Then,

1. v is an L_p-vertex if v is adjacent to a P-vertex,
2. v is an L_h-vertex if
   (a) v is not adjacent to any P-vertex,
   (b) v is adjacent to an H-vertex or a U-vertex, and
   (c) v can become happy (i.e., $|N^u(v)| + \max\{|N_i(v)|: 1 \le i \le k\} \ge q$),
3. v is an L_u-vertex if
   (a) v is not adjacent to any P-vertex, and
   (b) v is destined to be unhappy (i.e., $|N^u(v)| + \max\{|N_i(v)|: 1 \le i \le k\} < q$),
4. v is an L_f-vertex if
   (a) v is not adjacent to any colored vertex, and
   (b) v can become happy.

One can verify that the subtypes in Definition 11 really form a partition of all L-vertices. Note that the L_u-vertex not only refers to the destined-to-be-unhappy L-vertex that is adjacent to an H-vertex or a U-vertex (like the L_u-vertex in MHV and the L_u-vertex in SoftMHV), but also refers to the destined-to-be-unhappy L-vertex that is not adjacent to any colored vertex, as discussed before Definition 11.

Below is the subset-growth approximation algorithm Growth-HardMHV for the HardMHV problem.

Algorithm GROWTH-HARDMHV
Input: A connected undirected graph G, a partial coloring function c, and an integer $q > 0$.
Output: A total vertex coloring for G.

1. $\forall 1 \le i \le k$, $V_i \leftarrow \{v : c(v) = i\}$.
2. **while** there exist L-vertices **do**
   (a) **if** there exists a P-vertex v **then**
      i. $i \leftarrow c(v)$.
      ii. Add its any $q - |N^s(v) \cap V_i|$ L_p-neighbors to V_i. The types of all affected vertices (including v and vertices in $N^2(v)$) are changed accordingly.
   (b) **elseif** there exists an L_h-vertex v **then**
      i. Let V_i be the vertex subset in which v has the maximum colored neighbors.
      ii. Add vertex v and its any $q - |N^s(v) \cap V_i|$ L-neighbors to V_i. The types of all affected vertices (including v and vertices in $N^2(v)$) are changed accordingly.
   (c) **else**
      Comment: There must be an L_u-vertex.
      i. Let v be any L_u-vertex. If v has colored neighbors, then let V_i be any vertex subset containing a colored neighbor of v. Otherwise let V_i be V_1.
      ii. Add vertex v to subset V_i. The types of all affected vertices (including v and vertices in $N(v)$) are changed accordingly.
   (d) **endif**
6. **endwhile**
7. **return** the vertex coloring $(V_1, V_2, \cdots, V_k)$.

Lemma 17. $|L_u^{new}| \le O(\Delta^2)|H^{new}|$.

Proof. The proof of the lemma is similar to that of Lemma 14. Only one point needs to pay attention. When the algorithm processes an L_u-vertex, there are only L_u-vertices or L_f-vertices (if any) in the current graph. Each time Algorithm GROWTH-HARDMHV processes an L_u-vertex v, it processes only *one* such vertex. So, if v has an L_f-neighbor u, u will become an L_h-vertex after the processing. This means that coloring an L_u-vertex does not generate any new L_u-vertex. We omit the other details of the proof. □

Theorem 18. *The HardMHV problem can be approximated within a factor of* $\Omega(\Delta^{-3})$ *in polynomial time.*

Proof. Each time an H-vertex is generated, at most q L-vertices are consumed (i.e., colored). So, for the number of newly generated H-vertices we have $|H^{new}| \ge (|L^{org}| - |L_u^{org}| - |L_u^{new}|)/q$. By Lemma 17, and noticing that $q \le \Delta$, we get

$$|H^{new}| \ge \frac{|L^{org}| - |L_u^{org}|}{O(\Delta^2)}. \tag{A.2.1}$$

Let *OPT* be the number of happy vertices in an optimal solution to the problem. By the same reason as in Lemma 4, we obtain

$$OPT \le |H^{org}| + (\Delta + 1)(|L^{org}| - |L_u^{org}|). \tag{A.2.2}$$

Let *SOL* be the number of happy vertices found by Algorithm GROWTH-HARDMHV. Then we have $SOL = |H^{org}| + |H^{new}| = \Omega(\Delta^{-3})OPT$ by the above two inequalities. As Algorithm GROWTH-HARDMHV obviously runs in polynomial time, the theorem follows. □

A.3. NP-hardness of HardMHV

Theorem 19. *The HardMHV problem is NP-hard for any constant* $k \ge 3$*, where* k *is the color number in the problem.*

Proof. We prove the theorem by reducing k-MHE (see Theorem 10) to HardMHV.

Given an instance (G, c) of k-MHE, we construct an instance (G', c', q) of HardMHV as follows. For each edge $(u, v) \in E(G)$, do the following. Add a vertex x_{uv} and $\Delta - 1$ vertices $y_1^{uv}, y_2^{uv}, \cdots, y_\Delta^{uv}$, where Δ is the maximum vertex degree of G. The vertices y_i^{uv}'s are called *satellite vertices*. Replace edge (u, v) by two edges (u, x_{uv}) and (x_{uv}, v). Connect each vertex y_i^{uv} to x_{uv} via an edge (x_{uv}, y_i^{uv}). Finally, let $q = \Delta + 1$. We thus get our HardMHV instance (G', c', q).

Since $q = \Delta + 1$, each original vertex $v \in V(G)$ and each newly added satellite vertex cannot be happy no matter how the vertices in G' are colored. For each edge $(u, v) \in E(G)$, since its corresponding vertex x_{uv} is of degree $\Delta + 1$, x_{uv} is happy iff its two neighbors u and v have the same color. This shows that the optimum of (G, c) is equal to that of (G', c', q), finishing the proof of the theorem. □

References

[1] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth and Brooks, Monterey, CA, USA, 1984.
[2] Gruia Calinescu, Howard Karloff, Yuval Rabani, An improved approximation algorithm for multiway cut, J. Comput. System Sci. 60 (3) (2000) 564–574.
[3] Elias Dahlhaus, David Johnson, Christos Papadimitriou, Paul Seymour, Mihalis Yannakakis, The complexity of multiterminal cuts, SIAM J. Comput. 23 (1994) 864–894.
[4] David Easley, Jon Kleinberg, Networks, Crowds, and Markets: Reasoning About a Highly Connected World, Cambridge University Press, 2010.
[5] Péter Erdös, László Székely, Evolutionary trees: an integer multicommodity max-flow-min-cut theorem, Adv. in Appl. Math. 13 (4) (1992) 375–389.
[6] Shimon Even, Robert E. Tarjan, Network flow and testing graph connectivity, SIAM J. Comput. 4 (1975) 507–518.
[7] Alan Frieze, Mark Jerrum, Improved approximation algorithms for max k-cut and max bisection, Algorithmica 18 (1997) 67–81.
[8] Satoru Iwata, Lisa Fleischer, Satoru Fujishige, A combinatorial strongly polynomial algorithm for minimizing submodular functions, J. ACM 48 (4) (2001) 761–777.
[9] Viggo Kann, Sanjeev Khanna, Jens Lagergren, Alessandro Panconesi, On the hardness of approximating max k-cut and its dual, Chic. J. Theoret. Comput. Sci. 1997 (1997).
[10] David Karger, Philip Klein, Clifford Stein, Mikkel Thorup, Neal Young, Rounding algorithms for a geometric embedding of minimum multiway cut, Math. Oper. Res. 29 (3) (2004) 436–461.
[11] Jon Kleinberg, Éva Tardos, Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields, J. ACM 49 (5) (2002) 616–639.
[12] Angsheng Li, Jiankou Li, Yicheng Pan, Pan Peng, Homophyly law of networks: principle, method and experiments, manuscript, 2012.
[13] Angsheng Li, Jiankou Li, Yicheng Pan, Pan Peng, Small community phenomenon in networks: mechanisms, roles and characteristics, manuscript, 2012.
[14] Angsheng Li, Pan Peng, Community structures in classical network models, Internet Math. 7 (2) (2011) 81–106.
[15] Angsheng Li, Pan Peng, The small community phenomenon in networks, Math. Structures Comput. Sci. 22 (3) (2012) 373–407.
[16] H. Saran, V. Vazirani, Finding k-cuts within twice the optimal, SIAM J. Comput. 24 (1995) 101–108.
[17] Thomas Schelling, Dynamic models of segregation, J. Math. Sociol. 1 (1972) 143–186.
[18] Thomas Schelling, Micromotives and Macrobehavior, Norton, 1978.
[19] Liang Zhao, Hiroshi Nagamochi, Toshihide Ibaraki, Greedy splitting algorithms for approximating multiway partition problems, Math. Program. 102 (1) (2005) 167–183.