The Eighth International Symposium on Operations Research and Its Applications (ISORA'09)
Zhangjiajie, China, September 20–22, 2009

Approximating Bounded Degree Maximum Spanning Subgraphs*

Wangsen Feng[1,†] Hao Ma[1] Bei Zhang[1]
Hanpin Wang[2]

[1]Key Laboratory of Network and Software Security Assurance, Ministry of Education
Computing Center, Peking University, Beijing 100871, P.R. China
[2]School of EECS, Peking University, Beijing 100871, P.R. China

Abstract The bounded degree maximum spanning subgraph problem arising from wireless mesh networks is studied here. Given a connected graph G and a positive integer $d \geq 2$, the problem aims to find a maximum spanning subgraph H of G with the constraint: for every vertex v of G, the degree of v in H, $d_H(v)$, is less than or equal to d. Here, a spanning subgraph is a connected subgraph which contains all the vertices of the original graph. We propose polynomial time approximation algorithms for cardinality case and edge weighted case respectively. When input graphs are edge unweighted, a 2-approximation algorithm is designed. When input graphs are edge weighted, the designed algorithm always outputs a spanning subgraph whose maximum degree is no more than $d+1$ and weight is at least $\frac{OPT(G)}{d+2}$, where $OPT(G)$ is the weight of optimal solutions. The bounded degree spanning subgraph output by the algorithm can be used as a transport subnetwork in wireless mesh networks.

1 Introduction

The problems of finding a maximum (minimum) subgraph satisfying certain constraints, such as the maximum matching problem, the maximum b-matching problem and the bounded degree minimum spanning tree problem, are intensively studied in combinatorial optimization and graph theory.

In this paper, we will discuss a novel computational problem arising from wireless mesh networks. The problem is named "Bounded degree maximum spanning subgraph problem" and defined as follows:

Bounded degree maximum spanning subgraph problem: Given a connected graph G and a fixed integer $d \geq 2$, ask for a maximum spanning subgraph H of G with $\Delta(H)$ no more than d, where a spanning subgraph is a connected subgraph which contains all the vertices of G.

If d is restricted to be 2, the problem is equivalent to the problem of finding a Hamilton path(cycle) in G. Thus, the problem is NP-Hard and we will focus on approximation algorithms for it.

*Supported by the Fundamental Research Program of China 973 under Grant No. 2009CB320505.
†Corresponding to fengws@pku.edu.cn

Let $OPT(G)$, $ALG(G)$ denote the number of edges in optimal solutions and solutions output by our algorithms respectively.

1.1 Motivation

Wireless mesh network is a network implemented over a wireless LAN, and its infrastructure type is decentralized, relatively inexpensive, reliable and resilient since the wireless routers are often static and not powered by batteries. In addition, the access points (routers) seldom change their positions in wireless mesh networks and some wireless nodes have gateway functions to provide the connectivity to Internet. Therefore, such networks behave almost like wired networks with infrequent topology changes and limited node failures. Due to these good properties and potential applications such as providing commercial Internet access to residents and local business, wireless mesh networks have drawn a lot of attention in recent years. For example, wireless mesh networks are being used as the last mile for extending the Internet connectivity for mobile nodes. In addition, some commercial deployments of multi-hop wireless meshes are already in the works.

Unlike traditional wired networks, close-by mesh routers share certain wireless channel to communicate and the throughput of the whole network is severely limited by wireless interference. The capacity of wireless mesh networks will be increased tremendously when extending from a single channel model to a multi-radio, multi-channel model. Usually, people design effective channel assignment, link scheduling, routing and transport subnetwork selection algorithms to reduce wireless interference [1, 2, 3, 4].

Let us consider the following special case. Given a wireless mesh network and d non-overlapping channels, suppose each mesh router in the network equips d network interfaces. Thus a connected d-edge coloring subnetwork can be chosen to transport data, since all the adjacent links in this subnetwork can use different channels to reduce wireless interference and increase network throughput.

For edge coloring, Vizing [5] states that for any graph G, either $\chi^{'}(G) = \Delta(G)$ or $\chi^{'}(G) = \Delta(G)+1$, where *chromatic index* $\chi^{'}(G)$ is the minimum number of colors needed in an edge coloring of G. Based on the statement, we can find a connected subgraph with maximum degree no more than $d-1$ in G and this subgraph must be d-edge colorable. Thus the original problem can be solved by finding a transport subnetwork with maximum degree no more than $d-1$ given fixed network topology and d non-overlapping channels.

1.2 Related Work

Bounded degree minimum spanning tree problem: Given a weighted graph $G = (V,E,c)$ and a positive integer $k \geq 2$, ask for a minimum cardinality (weight) spanning tree with maximum degree no more than k, where c is a weighted function defined on E, $c : E \rightarrow R^{+}$.

Clearly, the problem is *NP*-hard. In 1991, Michel X. Geomans proposed the following conjecture.

Conjecture: In polynomial time, one can find a spanning tree of maximum degree no more than $k+1$ whose cost is at most $OPT(k)$, the minimum cost of any spanning tree of maximum degree no more than k.

1. **Input:** a connected graph $G = (V,E)$ and a integer $d \geq 3$
2. **Output:** a spanning subgraph H_{ALG} with maximum degree no more than d

3. Compute a spanning tree T in G with $\Delta(T) \leq d$;
4. Define a function $f(v) : V \to Z$, for any vertex v in V, $f(v) = d - d_T(v)$;
5. Compute a maximum f-matching M_f in the residual graph $G' = G - T$ with $d_{M_f}(v) \leq f(v)$;
6. Let $H_{ALG} = T \cup M_f$, output H_{ALG}.

Figure 1: Algorithm 1

Michel X. Geomans [6] proves that one can find a spanning tree of maximum degree no more than $k+2$ whose cost is at most $OPT(k)$ in polynomial time. In 2007, Mohit Singh and Lap Chi Lau [7] proved the conjecture. The algorithms proposed in [6, 7] are all based on linear programming.

2 Approximation algorithms based on bounded degree spanning trees

In this section, approximation algorithms for bounded degree maximum spanning subgraph will be designed. The main idea is natural. First, we construct a bounded degree spanning tree to guarantee the connectivity and then try to add as many edges as possible to the tree keeping the degree bound at the same time.

The problem of finding bounded degree minimum spanning tree has been well studied and an approximation algorithm based on linear programming has been designed. We will make use of a bounded degree spanning tree to construct the backbone of transport subnetworks and maximize the number of links in the subnetworks by maximum b-matchings to increase the throughput of wireless mesh networks. A maximum b-matching is a natural generalization of a maximum matching. Given a graph $G = (V,E)$ and an integer function $b(v) : V \to Z^+$, if M_b is a subset of edge set E and satisfying that for every vertex v of G, the degree of v in the subgraph induced by M_b is less than or equal to $b(v)$, then M_b is called a b-matching of G. A maximum b-matching is a b-matching with maximum number of edges among all the b-matchings. In 1983, Gabow [8] designed an algorithm of complexity $O(|V||E|\log|V|)$ for finding a maximum b-matching.

Corresponding to the cases of link bandwidth equal to each other or not, we will design approximation algorithms for cardinality case and weighted case respectively.

2.1 Cardinality case

There exists a polynomial time algorithm to find a spanning tree of maximum degree at most $k+1$ in a graph with a spanning tree of maximum degree k. Here, we assume that the fixed integer $d \geq \Delta(T)+1$, where T is a spanning tree of G with the least maximum degree. Thus, $d \geq 3$.

Fig. 2-4 give a demo of Algorithm 1. The input instance is a connected graph with 10 vertices and 19 edges, and the degree bound d is 3.

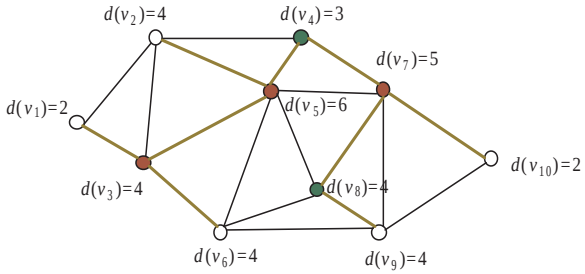

Figure 2: (3) compute $T : \Delta(T) \leq 3$. The bold edges are edges in T.

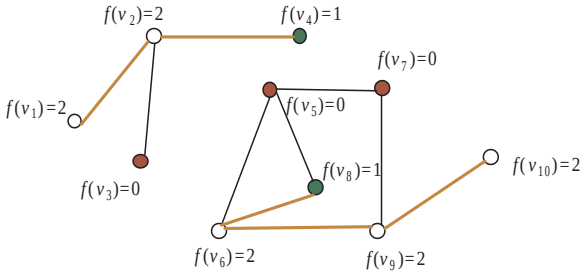

Figure 3: (4,5) compute $M_f : d_{M_f}(v) \leq f(v)$. The bold edges are edges in M_f.

Now, we analyze the time complexity of Algorithm 1. In the first step, the spanning tree with degree bound d can be constructed in $O(|V||E|\log|V|\alpha(|V|))$ time by employing the approximation algorithm proposed in [9], where α is inverse Ackerman function. The second step costs $O(|V|)$ time. In the third step, a maximum b-matching can be found in $O(|V||E|\log|V|)$ time. The fourth step costs $O(|V|)$ time. As a consequence, the time complexity of Algorithm 1 is $O(|V||E|\log|V|\alpha(|V|))$. Then we discuss the approximation ratio of Algorithm 1.

Theorem 1. *For any connected graph G, the approximation ratio of Algorithm 1 is 2.*

Proof: Let H_{OPT} be an optimal solution, thus H_{OPT} is a maximum spanning subgraph of G with maximum degree no more than d. Let $H_{ALG} = T \cup M_f$ be the subgraph output by Algorithm 1 and n be the number of vertices in G.

Based on Algorithm 1, H_{ALG} is a maximum subgraph of G with the degree bound and T being its subgraph. Thus, for any subgraph H with $\Delta(H) \leq d$ and T being its subgraph, $|E(H)| \leq |E(H_{ALG})|$. If we add all the edges which are not in H_{OPT} but in T into H_{OPT} and delete some edges in H_{OPT} to keep the degree bound at the same time, then an instance of H can be obtained.

Without loss of generality, suppose there are $x(1 \leq x \leq n-1)$ edges in T not in H_{OPT}. When adding one of the x edges into H_{OPT}, it is easy to see that only the degrees of the two endpoints of the edge are affected and each of them increased by one. There are three cases.

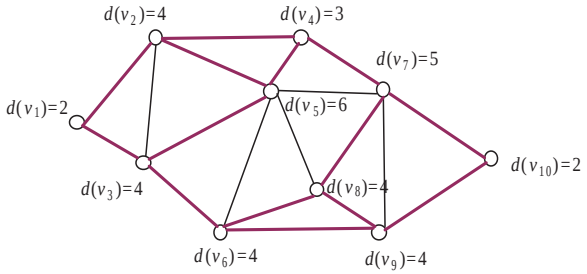

Figure 4: (6) output $H_{ALG} = T \cup M_f$. The bold edges are edges in H_{ALG}.

1. the degrees of both endpoints are still bounded by d: no edges in H_{OPT} need to be deleted.

2. the degree of one of the two endpoints is greater than d: just delete one edge in H_{OPT} not in T incident to the endpoint to keep the degree bound.

3. the degrees of both endpoints are greater than d: it means after the edge is added into H_{OPT}, the degrees of its two endpoints in H_{OPT} are increased to $d+1$. For each endpoint, just delete one edge in H_{OPT} not in T incident to it to keep the degree bound.

It is easy to see at each step, the number of edges in H_{OPT} decreases by one at most. Thus, $|E(H)| \geq |E(H_{OPT})| - x \geq |E(H_{OPT})| - n + 1$. On the other hand, $|E(H_{ALG})| \geq |E(H)|$, thus $|E(H_{ALG})| \geq |E(H_{OPT})| - n + 1$ and $|E(H_{OPT})| \leq |E(H_{ALG})| + n - 1$. Clearly, $|E(H_{ALG})| \geq |T| = n - 1$. As a consequence,

$$\frac{OPT(G)}{ALG(G)} \leq \frac{|E(H_{ALG})| + n - 1}{|E(H_{ALG})|} = 1 + \frac{n-1}{|E(H_{ALG})|} \leq 2 \tag{1}$$

□

2.2 Weighted case

Let $G = (V, E, c)$ be an edge weighted graph, where c is a cost function defined on the edge set E, $c(e) : E \to R^+$. Denote by $c(e)$ the cost of an edge e. Let $c(H) = \sum_{e \in E(H)} c(e)$ be the cost of H, where H is a subgraph of G.

In this case, in order to reuse the framework of Algorithm 1, we need to construct a bounded degree **maximum** spanning tree. We will construct the tree based on the algorithm of finding bounded degree minimum spanning trees. Let $G = (V, E, c)$ be a weighted graph, where $c : E \to R^+$ is a weight function defined on E. Let w be the maximum weight of the edges in G, $w = \mathrm{MAX}_{e \in E} c(e)$. Construct a weighted graph $G' = (V, E, c')$, where $c' : E \to R^+$ and $c'(e) = w + 1 - c(e)$. Clearly, a bounded degree maximum spanning tree in G corresponds to a bounded degree minimum spanning tree in G'.

On the other hand, there exists a polynomial time algorithm to find a spanning tree of maximum degree no more than $k+1$ and with its cost at most $OPT(k)$, where $OPT(k)$ is

1. **Input:** a edge weighted connected graph $G = (V, E, c)$ and a integer $d \geq 2$
2. **Output:** a spanning subgraph H_{ALG} with maximum degree no more than d+1

3. Compute a spanning tree T in G with $\Delta(T) \leq d+1$ and $c(T) \geq c(T_d)$, where T_d is a maximum weight spanning tree with $\Delta(T_d) \leq d$;
4. Define a function $f(v) : V \to Z$, for any vertex v in V, $f(v) = d + 1 - d_T(v)$;
5. Compute a maximum f-matching M_f in the residual graph $G' = G - T$ with $d_{M_f}(v) \leq f(v)$;
6. Let $H_{ALG} = T \cup M_f$, output H_{ALG}.

Figure 5: Algorithm 2

the minimum cost of any spanning tree of maximum degree no more than k. Thus, we can compute a spanning tree of maximum degree no more than $k+1$ and with its cost at least $OPT'(k)$, where $OPT'(k)$ is the maximum weight of any spanning tree with maximum degree no more than k.

Based on the discussions above, we get Algorithm 2 for weighted case.

Theorem 2. *For any edge weighted connected graph G, Algorithm 2 returns a spanning subgraph H_{ALG} in G with $\Delta(H_{ALG}) \leq d+1$ and $c(H_{ALG}) \geq \frac{OPT(G)}{d+2}$, where $OPT(G)$ is the weight of optimal solutions.*

Proof: Let H_{OPT} be an optimal solution, thus H_{OPT} is a maximum spanning subgraph of G with maximum degree no more than d. Let $H_{ALG} = T \cup M_f$ be the subgraph output by Algorithm 2. Let T_{OPT} be a maximum spanning tree in H_{OPT}. Clearly, the maximum degree of T_{OPT} is no more than d, thus $c(T_{OPT}) \leq c(T_d) \leq c(T)$.

Based on Algorithm 2, H_{ALG} is a maximum subgraph of G with $\Delta(H_{ALG}) \leq d+1$ and T being its subgraph. Thus, for any subgraph H with $\Delta(H) \leq d+1$ and T being its subgraph, $c(H) \leq c(H_{ALG})$. Similarly, to obtain an instance of H, we can add all the edges which are not in H_{OPT} but in T into H_{OPT} and delete some edges to keep the degree bounded $d+1$.

Now we describe how to delete edges. For each vertex v in V, $d_T(v) - 1$ edges need to be deleted at most in order to keep the degree bounded $d+1$. Clearly, the edges incident to v in H_{OPT} not in T are either in T_{OPT} or in $H_{OPT} - T_{OPT}$.

If such an edge e belongs to $H_{OPT} - T_{OPT}$, then $c(e)$ is less than or equal to $c(e_v)$, where e_v is the edge between v and its parent in T_{OPT}, since T_{OPT} is a maximum spanning tree in H_{OPT}. Thus the cost of the edges incident to v in $H_{OPT} - T_{OPT}$ to be deleted must less than or equal to $d \times c(e_v)$. Considering all the vertices, we know that the total cost of the edges in $H_{OPT} - T_{OPT}$ to be deleted is no more than $d \times c(T_{OPT})$.

On the other hand, the sum of weights of edges need to be deleted in T_{OPT} is less than or equal to $c(T_{OPT})$.

Thus, $c(H_{OPT}) - (d+1) \times c(T_{OPT}) \leq c(H) \leq c(H_{ALG})$ and $c(H_{OPT}) \leq c(H_{ALG}) + (d+1) \times c(T_{OPT})$. Clearly, $c(T_{OPT}) \leq c(T) \leq c(H_{ALG})$. As a consequence:

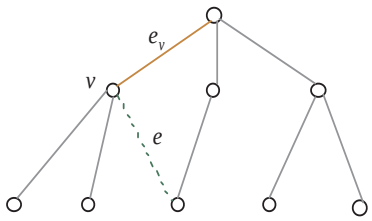

Figure 6: $c(e) \leq c(e_v)$

$$\begin{aligned} \frac{OPT(G)}{ALG(G)} &\leq \frac{c(H_{ALG}) + (d+1) \times c(T_{OPT})}{c(H_{ALG})} \\ &= 1 + \frac{(d+1) \times c(T_{OPT})}{c(H_{ALG})} \\ &\leq d+2 \end{aligned} \tag{2}$$

□

References

[1] M. Kodialam and T. Nandagopal. Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem. MobiCom 2003: 42-54.

[2] M. Alicherry, R. Bhatia and L.E. Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. MobiCom 2005: 58-72.

[3] Weizhao Wang, Yu Wang, Xiang-Yang Li, Wenzhan Song and Ophir Frieder. Efficient interference-aware TDMA link scheduling for static wireless networks. MobiCom 2006: 262-273.

[4] Jie Gao, Leonidas J. Gubas, John Hershberger, Li Zhang and An Zhu. Geometric spanners for routing in mobile networks. IEEE Journal on Selected Areas in Communications 23(1): 174-185, 2005.

[5] Vizing V.G. On an estimate of the chromatic class of a p-graph. (in Russian) Diskret. Analiz. 3: 25-30, 1964.

[6] Michel X. Goemans. Minimum bounded degree spanning trees. FOCS 2006, 273-282.

[7] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. STOC 2007: 661-670.

[8] H.N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. STOC 1983: 448-456.

[9] Martin Fürer and Balaji Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. SODA 1992: 317-324.