Topology Formation for Wireless Mesh Network Planning

Chun-cheng Chen*, Chandra Chekuri*, and Diego Klabjan†
*Dept of Computer Science, Univ of Illinois at Urbana Champaign
†Dept of Industrial Engineering and Management Sciences, Northwestern University

***Abstract*—In this paper, we propose Greedy Selection Rounding (GSR), an efficient and near-optimal algorithm to design a wireless mesh network topology that maximizes the coverage of the users while ensuring that the network is resilient to node failures and and the deployment cost is under a given budget. In the case that GSR fails to find a solution satisfying the budget constraint, the incurred cost does not exceed the budget by a constant factor. Through extensive evaluation, we show that in all our test cases GSR always generates a topology above 95% of the optimal in terms of the number of covered users while never exceeding the budget by more than 15%.**

I. Introduction

Planning wireless mesh networks is difficult. Most of the existing works [2], [3], [10], [12] focus on the interference alleviation aspect of the planning problem through antenna selection, power/channel assignment, or measurement-based link quality estimation. However, aspects of the network design problem including resilience to failures, cost, and coverage are ignored. The difficulty of the wireless mesh network planning problem lies in the inter-dependency between the different components of the problem (see Figure 1). Modeling and simultaneously optimizing all aspects of the planning problem is extremely difficult. Consequently, it is common for people [2], [3], [4], [10], [11], [12] to focus on only one aspect of the problem, i.e. interference mitigation. Such an imbalance in the importance given to network resiliency and interference mitigation could be catastrophic. In fact, during the San Diego wildfire in October 2007, two of the network towers of TDV[1] were consumed and the service halted.

We propose to address the inter-dependency between the planning components by decoupling the planning problem into two parts–we first address topology formation and then address interference mitigation, as shown in Figure 1. In topology formation, we (mostly) care about forming a well-connected and cost-efficient network topology that maximizes the number of covered users. Once the topology is formed, the interference can be mitigated as is done in [9], [10], [12]. Such a two-phase approach greatly reduces the complexity of the wireless mesh network planning problem, allowing us to find a good solution in a reasonable time.

In this paper, we focus on the topology formation of wireless mesh network planning. We envision that there will be some potential locations for placing the wireless mesh routers. Each location covers some number of users that may access the mesh network. For example, such locations in a city could be public buildings, lamppoles, or traffic signals. Some of the locations are so *important* that routers have to be deployed at these locations. Furthermore, we assume that the gateway locations are pre-determined. We believe this is a realistic assumption because (i) some locations may already have existing connectivity and thereby can serve as gateways directly; (ii) for performance considerations, it is reasonable to place the gateways in a structured way to avoid forming long hops in the backbone. Our modeling is guided by three criteria for a good network topology shown as follows.

We would like to thank Nitish John Korula, Jin Suk Kim, Fang-Kai Jao, and Zheng Zeng for their helpful comments and the fruitful brainstorming discussions. This research is supported in part by Vodafone Fellowship.

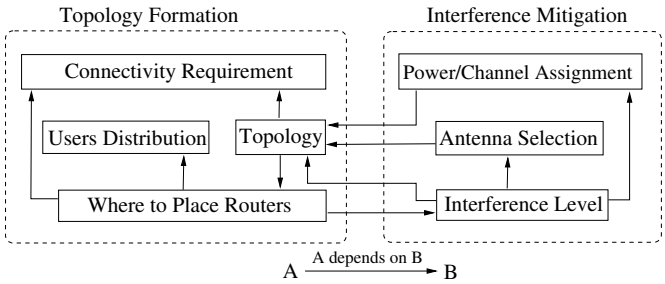

Fig. 1. Dependency diagram for wireless mesh network planning

Robustness: Once a router is deployed at a location, it is required that there are at least two node disjoint paths connecting it to the gateways. This ensures that any single node failure does not disconnect the network.

Budget Constraint: We assume that deploying a router at a location incurs a fixed cost. A given budget constraint on the total cost translates into a bound on the number of locations that can be installed with routers.

User Coverage: We assume each location covers some number of users and each user may be covered by several locations. Note that a user is counted *only once* even though he or she may be able to connect via multiple locations.

Our goal is to find a topology so that (1) the number of users is covered as many as possible (2) the robustness and budget constraints are satisfied. In this paper, we propose GSR, an efficient algorithm that delivers a near optimal solution to the Topology Formation Problem (TFP). GSR always finds a topology satisfying the connectivity requirement if there exists one. In the case that GSR fails to find a topology respecting the budget constraint, it returns the topology whose incurred cost is within a constant factor of the budget – a fail-safe property. To the best of our knowledge, we are the first to study the topology formation for wireless mesh network planning under the condition of user coverage, budget constraints, and robustness requirements.

The rest of the paper is organized as follows. We describe

our network model in Section II, present GSR in Section III. We further discuss GSR's fail-safe property in Section IV, relax the assumptions in Section V, evaluate GSR in Section VI, and finally conclude the work in Section VII.

II. Problem Formulation

We formulate TFP as follows. Let $G = (V, E)$ be an undirected graph. V represents the set of all gateway locations, required important locations, and candidate locations for wireless mesh router placement. There is an edge $(u, v) \in E$, $u, v \in V$, if the distance between u and v is within the communication range of each other. Let $GW \subset V$ be the set of pre-determined gateway locations. Also, let $R \subset V$ be the set of important locations required to have router placement. When a location $u \in V - GW$ is deployed with a router, there must be at least two vertex disjoint paths from u to the gateways.

Due to the budget constraint, we are allowed to install at most a certain number of routers, say κ. For ease of exposition, we assume each candidate location $v \in V$ is associated with a profit $w(v)$, representing the number of users covered by location v. This assumption will be relaxed later.

The goal is to find a subgraph $H \subseteq G$ such that (1) there are at least 2 vertex disjoint paths connecting each location deployed with a router to the gateways, (2) the number of installed mesh routers is at most κ, and (3) the total profit of H is maximized. We formally define TFP as follows:

Definition *1 (***TFP***): The input consists of an undirected graph $G = (V, E)$, a set $GW \subset V$ of gateways, a set $R \subset V$ of required nodes, a budget $\kappa \in \mathbf{N}$, and for each vertex $v \in V$ an associated profit $w(v) \geq 0$. The objective is to find the max-profit subgraph $H \subseteq G$ so that $GW \subset V(H)$, $R \subset V(H)$, $|V(H)| \leq \kappa$, and each $v \in V(H) - GW$ has two vertex disjoint paths to gateways.*

Remark 1: Without loss of generality we can merge all the pre-determined gateway nodes into one virtual super node. The problem remains the same in terms of feasibility and cost.

III. Topology Formation

In this section, we discuss the hardness of TFP and propose the approximation algorithm GSR.

A. Hardness

TFP is a fairly complicated problem. Its difficulty lies in the fact that TFP has the dual objectives of (1) maximizing the profit of nodes in $V(H)$ (2) keeping the number of used nodes below κ, while ensuring that all nodes in $V(H) - GW$ have the desired connectivity. These dual objectives make it hard to define a standard notion of approximation ratio. Consider the *simpler* problem where the profit of every node in $V(G) - R$ is zero – in other words, the only question is whether we can *connect* R to the gateways using at most κ nodes (κ includes the gateways). This is essentially the TFP-C problem that we define as follows.

Definition *2 (**TFP-C**): Given an undirected graph $G = (V, E)$ so that each vertex $a \in V$ is associated with a cost $c(a)$. Furthermore, there is a value $r_{u,v}$ for each pair $u, v \in V$. The objective is to find the min-cost subgraph of G so that there are $r_{u,v}$ vertex disjoint paths connecting u and v.*

As we remarked earlier, the gateway nodes can be merged into a single super node. Let us define α_v as the connectivity requirement for each $v \in R$. That is, $\alpha_v = r_{s,v}$ in TFP-C, where s is the super gateway node. Even when $\alpha_v = 1$ for each $v \in R$, this problem is equivalent to the node-weighted Steiner tree problem [7] and hence NP-hard. Note that the problem remains NP-hard even when $c(v) = 1$ for each v. It turns out that even having a good approximation algorithm for TFP-C is very difficult [5].

B. Greedy Selection Rounding

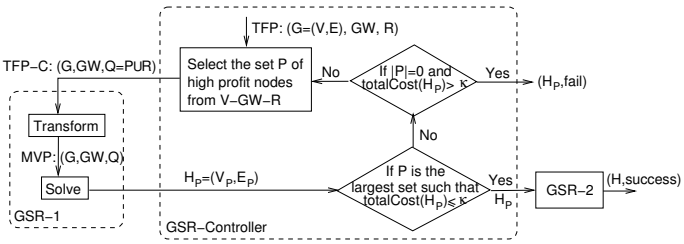

Fig. 2. Flow chart of GSR

Even for a simpler TFP-C, it is unlikely to exist an efficient good approximation algorithm. We therefore decompose the process of solving TFP in several stages. As shown in Figure 2, GSR consists of three components – GSR-Controller, GSR-1, and GSR-2. Given input $(G = (V, E), GW, R)$ for TFP, the job of GSR-Controller is to pre-select the set P of high profit nodes from $V - GW - R$, assign robustness requirement 2 to each of the selected nodes, combine the selected nodes in P and R so that $Q = P \cup R$, and pass to GSR-1 the input $(G = (V, E), GW, Q)$. Given the input from GSR-Controller, GSR-1 tries to find the min vertex cost subgraph of G satisfying the connectivity constraints of the set Q, which is equivalent to solving TFP-C. However, we have seen in Section III-A that a polynomial time approximation algorithm for TFP-C is unlikely to exist. Ideally, if we can transform the vertex-cost-based TFP-C into the edge-cost-based MVP defined below, then we will see in Section III-B1 that there is a 2-approximation algorithm.

Definition *3 (**MVP**):* Given an undirected graph $G = (V, E)$ *so that each edge $e \in E$ is associated with a cost $c(e)$. Furthermore, there is a value $r_{u,v}$ for each pair $u, v \in V$. The objective is to find the min-cost subgraph of G so that there are $r_{u,v}$ vertex disjoint paths connecting u and v.*

GSR-1 then transforms TFP-C to MVP by distributing the vertex costs to the edges. After finding the subgraph $H_P \subseteq G$ for the MVP, GSR-1 returns H_P to GSR-Controller. GSR-Controller checks if $|P|$ is the largest set such that the incurred cost of H_P is at most the budget κ. If the answer is yes, it passes H_P to GSR-2 for further processing. Otherwise, GSR-Controller adjusts the size of the pre-selected set P and repeats the pre-selection process again. If the total cost of H_P exceeds the budget κ without any node being pre-selected, i.e. the budget is too stringent just to satisfy the connectivity requirement of R, GSR-Controller returns H_P and declares failure to find a feasible solution. Given the input H_P from GSR-Controller, GSR-2 expands H_P by greedily including nodes with high profits while ensuring connectivity of each newly added node is maintained and the total cost is under the budget.

$$\min \sum_{e \in E(G)} c(e)x(e)$$

$$\text{s.t.} \sum_{e \in \delta(S,S')} x(e) \geq f_k(S,S') - |V - S - S'|$$

$$\forall S, S' \subset V, S \cap S' = \emptyset \quad (1)$$

$$0 \leq x(e) \leq 1 \qquad \forall e \in E(G) \quad (2)$$

Fig. 3. **(LP-MVP)** Linear program for the min-edge-cost vertex-connectivity problem

1) Preliminaries for Solving MVP: Recently, Fleischer et al. [6] proposed a linear programming formulation for MVP.

Let $\delta(S, S')$ be the set of edges connecting S and S', where S, S' are two disjoint subsets of V in a graph G. Formally speaking $\delta(S, S') = \{(u, v) \in E : u \in S, v \in S', S \subset V, S' \subset V, S \cap S' = \emptyset\}$. Let $f_k(S, S')$ be the largest connectivity requirement between S and S'. In other words, $f_k(S, S') = \max\{r_{u,v} | u \in S, v \in S'\}$, where $r_{u,v} \in \{0, 1, 2, ..., k\}$. Define $x(e) = 1$ if edge e is included in the solution subgraph of MVP; otherwise, $x(e) = 0$. Consider the linear program in Figure 3. It is easy to see that Condition 1 is necessary. It turns out that the condition is also sufficient.

Lemma *2 ([6]): The set of integral solutions to the linear program LP-MVP in Figure 3 equals the set of solutions to the min-edge-cost vertex-connectivity problem (MVP).*

In fact, LP-MVP possesses a nice property as follows.

Theorem *3 ([6]): Any basic solution to LP-MVP in Figure 3 has at least one variable e such that $x(e) \geq 1/2$ when $r_{u,v} \in \{0, 1, 2\}$, $u, v \in V$.*

Based on Lemma 2, Theorem 3, and the fact that there exists a polynomial-time separation oracle for LP-MVP, Fleischer et al. [6] proposed a 2-approximation algorithm for MVP. As shown in Figure 4, the solution set F of edges is initially empty. It first solves LP-MVP. From Theorem 3, the basic feasible solution must have at least one variable $x(e)$ with value at least $1/2$. The algorithm rounds up to 1 the $x(e)$ variable whose value is at least 1/2 and adds to F the edge $e = (u, v)$. The process of solving the LP and rounding up the variables is repeated until all the connectivity requirements $r_{u,v}$, where $u, v \in V$, are satisfied or every edge $e \in E(G)$ has been added to F. If the connectivity requirement for some $r_{u,v}$ can not be satisfied but all edges in $E(G)$ have been added to F, an exception indicating a bad input is raised.

Although LP-MVP can be solved in polynomial time *theoretically*, it is very inefficient to solve LP-MVP in practice due to the exponential number of constraints. One of our contributions is to formulate an alternative LP, called LP-MVP$_{GSR}$, *equivalent* to LP-MVP but only requiring a *polynomial* number of variables and constraints while still preserving the nice property of 2-approximation.

2) GSR-Controller: The main job of GSR-Controller is to pre-select high profit nodes. We use an example to illustrate GSR-Controller's binary search for the number of the pre-selected nodes:

Example: Given the graph $G = (V, E)$, suppose the budget $\kappa = 62$, $|GW| = 10$, $|R| = 20$, and $|V| = 100$. GSR-Controller first finds in $V - GW - R$ the set of nodes that have two vertex disjoint paths to the gateways. This can be easily done by finding the bi-connected components of graph G. Suppose there are 40 nodes in $V - GW - R$ that have

solve-MVP($G = (V, E)$)
1: $F \leftarrow \emptyset$
2: **while** there exists unsatisfied connectivity requirement $r_{u,v}$, $u, v \in V$ & there is some edge $e \in E(G)$ not added to F **do**
3: solve LP-MVP
4: **for all** $e \in E(G)$ and e is not added to F yet **do**
5: **if** $x(e) \geq 1/2$ **then**
6: $F \leftarrow F \cup e$
7: $x(e) \leftarrow 1$
8: **if** some connectivity requirement $r_{u,v}$ is not satisfied **then**
9: raise exception("bad input G")
10: return F

Fig. 4. Pseudo-code for solving MVP

two vertex disjoint paths to the gateways, it is easy to see that we cannot select all of these 40 nodes for otherwise there would be 40+10+20 = 70 nodes, exceeding the budget 62. Therefore GSR-Controller *greedily* chooses the top 32 high profit nodes as the set P of pre-selected nodes (ties are broken arbitrarily), assigns each of them connectivity requirement 2, and asks GSR-1 to solve the TFP-C. Suppose the cost of the returned graph H_{32} from GSR-1 exceeds 62, GSR-Controller then tries to pre-select the top 16 high profit nodes and determine if the cost of H_{16} is at most 62. Continuing this procedure, the final trace of the number of pre-selected nodes is $32(\times) \rightarrow 16(\times) \rightarrow 8(\surd) \rightarrow 12(\surd) \rightarrow 14(\times) \rightarrow 13(\surd)$. GSR-Controller then passes H_{13} to GSR-2 for further processing.

We note that when the connectivity requirement for Q is satisfied, each of the intermediate nodes connecting the nodes in Q to the gateways will also have two vertex disjoint paths to the gateways. Therefore, H_P satisfies the connectivity requirement of TFP.

3) GSR-1 and LP-MVP$_{GSR}$: GSR-1 transforms TFP-C to MVP as follows. For each vertex $v \in V$, GSR-1 distributes half of its cost to each of its incident edges. That is, for each edge $e = (u, v)$ we have $c(e) = 1/2 \cdot c(u) + 1/2 \cdot c(v)$. Since the costs are now on the edges, TFP-C becomes MVP. We next show our proposed compact formulation LP-MVP$_{GSR}$ that serves as the basis of the 2-approximation algorithm for MVP.

We create another set of directional edges $\vec{E}$ that correspond to the *directed* version of E. Therefore, each undirected edge $e = (u, v) \in E$ corresponds to two directed edges $\vec{e} = (u, v)$ and $\vec{e} = (v, u)$ in $\vec{E}$, where $|\vec{E}| = 2|E|$. We define the variables as follows:

$$f_{\vec{e},v} = \begin{cases} 1, \text{ if edge } \vec{e} \in \vec{E} \text{ carries flow originating from } v \in V \\ 0, \text{ otherwise} \end{cases}$$

$$z_{\vec{e}} = \begin{cases} 1, \text{ if edge } \vec{e} \in \vec{E} \text{ carries flow} \\ 0, \text{ otherwise} \end{cases}$$

$$y_e = \begin{cases} 1, \text{ if edge } \vec{e} = (u, v) \text{ or } \vec{e} = (v, u) \text{ carries flow, } \vec{e} \in \vec{E} \\ 0, \text{ otherwise} \end{cases}$$

A complete LP formulation is shown in Figure 5.

Since there must be at least 2 vertex disjoint paths from each node in Q to some gateways, the sum of the flows originating from $v \in Q$ must be at least 2, as is shown in Eq. (3). The technique we use to ensure vertex disjoint paths is that for a vertex u, there can be at most one incoming flow originating

$$\min \sum_{e \in E} c(e) y_e$$

$$\text{s.t.} \sum_{\vec{e}:\vec{e}=(v,u)\in\vec{E}} f_{\vec{e},v} \geq 2 \quad \forall v \in Q \tag{3}$$

$$\sum_{\vec{e}:\vec{e}=(w,u)\in\vec{E}} f_{\vec{e},v} \leq 1 \quad \forall u \in V - GW, \ \ \forall v \in Q \tag{4}$$

$$\sum_{\vec{e}:\vec{e}=(w,u)\in\vec{E}} f_{\vec{e},v} = \sum_{\vec{e}:\vec{e}=(u,w)\in\vec{E}} f_{\vec{e},v}$$
$$\forall u \in V - GW, \ \ \forall v \in Q, \ \ u \neq v \tag{5}$$

$$f_{\vec{e},v} \leq z_{\vec{e}} \quad \forall \vec{e} \in \vec{E}, \ \forall v \in Q \tag{6}$$

$$f_{\vec{e},v} = 0 \quad \forall v \in Q, \ \forall \vec{e} = (w,v) \in \vec{E} \tag{7}$$

$$z_{\vec{e}=(u,v)} = z_{\vec{e}=(v,u)} = y_e \quad \forall e = (u,v) \in E \tag{8}$$

$$0 \leq f_{\vec{e},v}, \ \ y_e, \ \ z_{\vec{e}} \leq 1 \tag{9}$$

Fig. 5. (**LP-MVP**$_{GSR}$) Linear program for MVP with polynomial number of constraints and variables.

from source $v \in Q$, as shown in Eq. (4). Eq. (5) is the flow conservation constraint. Eq. (6) is the capacity constraint. Eq. (7) further ensures that a flow originating from source v can not go back to v again, preventing a cycle forming at the source. Eq. (8) relates the undirected edge variable y_e with the capacity $z_{\vec{e}}$ of the directed edges.

The following theorems give some useful properties of the formulation. In this paper, we omit all the proofs due to space restrictions. Interested readers are referred to our technical report [5] for detailed proofs.

Theorem *4: When all the y_e variables in Figure 5 are assigned with integral values, in any basic solution, the flow variables $f_{\vec{e},v}$ are integral.*

Theorem *5: LP-MVP$_{GSR}$ in Figure 5 is equivalent to LP-MVP in Figure 3 in the following sense;*

- *for every feasible solution $\bar{x}$ to LP-MVP in Figure 3, there is a feasible solution $(\bar{y}, \bar{f}, \bar{z})$ to LP-MVP$_{GSR}$ in Figure 5 such that $x(e) = y(e)$ for each $e \in E$.*
- *similarly, for every feasible solution $(\bar{y}, \bar{f}, \bar{z})$, the solution $\bar{x}$ with $x(e) = y(e)$ for each $e \in E$ is feasible for LP-MVP in Figure 3.*

From Theorem 4 and 5, LP-MVP$_{GSR}$ possesses the nice properties of LP-MVP and a 2-approximation algorithm is readily available. Thus, MVP can now be practically and efficiently solved in polynomial time.

Once the MVP problem is solved, GSR-1 obtains the subgraph $H_P \subseteq G$ where $E(H_P)$ consists of all the edges in F and $V(H_P)$ is the set of corresponding endpoints of the edges in $E(H_P)$.

4) GSR-2: GSR-2 greedily expands H_P while ensuring the newly added nodes have two vertex disjoint paths to the gateways. GSR-2 works by finding in $G - H_P$ the set of nodes adjacent to H_P, where each node has at least two edges connecting to H_P. Let this set of nodes be NB. GSR-2 expands H_P by adding to $V(H_P)$ the node $u \in NB$ with the largest profit and adding to $E(H_P)$ edges (u, v) and (u, w), where $v, w \in V(H_P)$. If there are more than two edges connecting u to H_P, GSR-2 chooses arbitrary two. This greedy expansion is continued until the budget κ is reached or there is no node adjacent to H_P with at least two edges connecting to H_P. The final expanded graph H_P is the solution subgraph H for TFP.

Theorem *6: GSR-2 preserves the property that each node newly added to H_P has 2 vertex disjoint paths to the gateways.*

IV. Fail-safe Property of GSR

Since GSR-1 iteratively adds edges to the solution set F until all the connectivity constraints for the nodes in Q are satisfied, GSR always finds a solution satisfying the connectivity constraints if there exists one. However, it may violate the budget constraint.

Lemma *7: Let H be the graph returned when GSR-Controller fails to satisfy the budget constraint. We have $V(H) - c \leq E(H) < 2V(H)$, where c is the number of connected components in H.*

Theorem *8: When GSR-Controller fails to satisfy the budget constraint, it returns at most 5 κ nodes.*

Theorem 8 provides a desirable property for practical purposes. Many requirements in reality may not be stringent and they can be tolerated up to a certain amount. In the case that GSR-Controller fails to find a feasible solution satisfying the budget constraint, the incurred cost is at most 5κ.

V. Relaxing the Assumptions of TFP

In this section, we relax two assumptions of TFP – 2-connectivity requirements for the important nodes and independent profit functions.

A. Higher Robustness for Nodes in R

Recall that α_v is the connectivity requirement for each $v \in R$. TFP assumes that $\alpha_v = 2$ for all $v \in R$. If the nodes in R have connectivity requirements larger than 2, IP-TFP can be easily extended by simply replacing the connectivity requirement for each node $v \in R$ with a larger value. The same applies to LP-MVP$_{GSR}$. Note that we can also allow different nodes in R to have different robustness requirements. The algorithm for solving the MVP remains essentially the same.

B. Submodular Profit Functions

For ease of exposition, our TFP formulation assumes that the profit $w(v)$ is only dependent on v itself. This is relaxed after we introduce the property called *submodularity*.

Definition *4: A function $f : 2^V \rightarrow \mathbf{Z}^+$ is said to be submodular if $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for all $A, B \subseteq V$.*

An alternate and equivalent definition of submodularity is that $f(\{v\} \cup A) - f(A) \leq f(\{v\} \cup B) - f(B)$, where $B \subset A \subseteq V$, $v \in V - A$; in other words the marginal benefit of v to a set decreases as more elements are added to the set. This intuitively captures many settings. Consider the TFP setting where there is an underlying population of users that will be served by the deployment of the routers. Each user can be served by some subset of routers that are nearby. Here, the real profit of deploying a set A of routers is the total number of users that can be connected to the gateway through some router in A; note that a user is counted only once even though he or she may be able to connect via multiple routers in A. We can then define profit function $f(A)$ to be the number of users that will be connected if A is deployed and it is easy to see that f is a submodular function.

Our GSR can be modified to handle a submodular profit function as follows: we distribute the vertex costs to the edges the same as before. The only change is in choosing the pre-selected nodes P to maximize $f(P)$. Instead of simply choosing nodes by the decreasing order of their individual profit we choose them greedily to maximize f as follows. We initialize P to be empty. In each step we pick a node v to maximize $f(P+v)-f(P)$, that is the node v that would add the most profit according to f. We do this iteratively until we choose the required number of nodes in P. The rest of the details in GSR-1 remain the same once P is chosen. We can also modify the post-selection process in GSR-2 as above using f as a guide instead of choosing according to $w(v)$. It is known that the greedy algorithm gives a constant factor $(1-1/e)$ approximation to maximize a submodular function when subjected to a simple bound constraint on the number of elements (nodes) picked [8]. Thus we expect the pre and post selection steps to perform well even for submodular profit functions.

VI. Performance Evaluation

We implemented GSR in C++. The component GSR-1 is implemented with CPLEX 10.0 LP solver using dual simplex method. We have implemented GSR-1 that rounds up variables in two different schemes. In GSR-B, GSR-1 rounds up the variables with values at least 1/2 *all at once*. On the other hand, in GSR-NB, GSR-1 rounds up the variables with values 1. If there does not exist such a variable, it randomly rounds up one of the variables whose values are at least 1/2 before re-solving the LP-MVP$_{GSR}$.

We evaluate GSR's performance over (1) grid graphs to emulate that the potential locations are road intersections and (2) unit disk graphs where nodes are randomly generated and there is a link between node u and v if and only if their distance is within some threshold. Each unit disk graph is generated so that each node has around 3.5 neighbors. When generating the graphs, we randomly assign node profits between 1 and 10 to represent the level of user coverage. The gateways and required locations are randomly chosen and each required important node in $|R|$ has robustness requirements randomly chosen between 2 and 4.

Figures 6(a) and 6(b) show the effect of varying κ with respect to the normalized objective value returned by GSR for grid and random unit disk graphs, respectively. When budget κ is small, there can hardly exist a feasible solution. We therefore only report the results for $\kappa > 60$ for grid graphs and $\kappa > 45$ for unit disk graphs. Even though GSR-1 is no longer a 2-approximation algorithm when the robustness requirement exceeds 2, it is clearly seen that GSR still performs nearly optimal (with medians always above 95% of optimal).

We then randomly generate 10 scenarios for grid graphs and unit disk graphs under the stringent budget constraints, given that GSR fails to find a feasible solution. Figures 7(a) and 7(b) show the returned cost given that GSR fails to find a feasible solution. Apparently, even though the returned cost exceeds the corresponding budget, it is always within 115% of the budget. Thus, we believe the fail-safe property of GSR is mostly preserved even for high robustness requirements and applicable for practical scenarios.

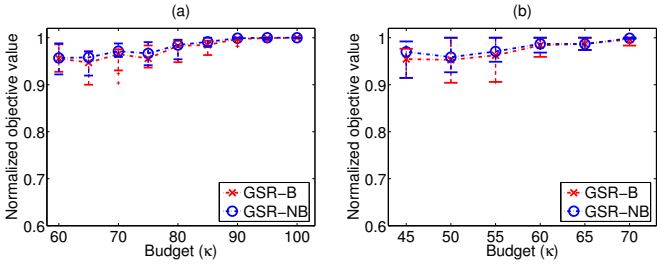

Fig. 6. Normalized objective value of GSR with varied κ for (a) grid graphs (b) unit disk graphs

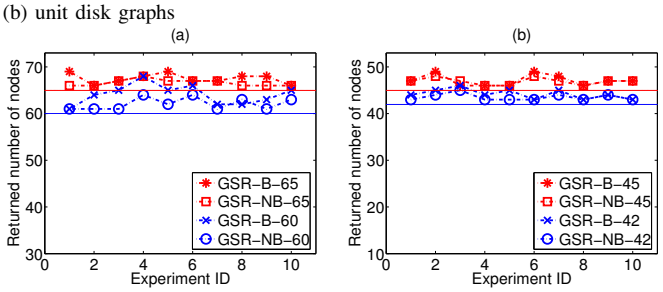

Fig. 7. Incurred cost given that GSR failed to find a feasible solution (a) at $\kappa = 60$ and $\kappa = 65$ for grid graphs (b) at $\kappa = 42$ and $\kappa = 45$ for unit disk graphs

VII. Conclusion

Most of the existing research effort on wireless mesh network planning focuses on the interference mitigation issue, assuming a given network topology. However, the problem of designing a good network topology that is resilient to failures, cost-effective, and maximizes the number of served users, has not received much attention. In this paper, we fill this void. We propose GSR, an efficient and near optimal algorithm for providing high user coverage (profits), while ensuring the network is robust to node failures and the deployment cost is under the budget. We have shown through evaluation that GSR is consistently near-optimal under various parameter settings.

References

[1] TDV. http://www.sctdv.net/.

[2] P. Bhagwat, B. Raman, and D. Sanghi. Turning 802.11 inside-out. In *Proceedings of HotNets-II*, 2003.

[3] J. Camp, J. Robinson, C. Steger, and E. Knightly. Measurement driven deployment of a twotier urban mesh access network. In *Proceedings of MobiSys*, 2006.

[4] R. Chandra, L. Qiu, K. Jain, and M. Mahdian. Optimizing the placement of internet taps in wireless neighborhood networks. In *Proceedings of IEEE ICNP*, 2004.

[5] C. Chen, C. Chekuri, and D. Klabjan. Topology formation for wireless mesh network planning. http://www.ews.uiuc.edu/~chen35/papers/topoform-tech-report.pdf.

[6] L. Fleischer, K. Jain, and D. P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *J. Comput. Syst. Sci.*, 72(5):838–867, 2006.

[7] P. N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995.

[8] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set function. *Math. Programming*, 14:265–294, 1978.

[9] B. Raman. Channel allocation in 802.11-based mesh networks. In *Proceedings of INFOCOM*, 2006.

[10] B. Raman and K. Chebrolu. Revisiting MAC design for an 802.11based mesh network. In *Proceedings of HotNets-III*, 2004.

[11] J. Robinson, M. Uysal, R. Swaminathan, and E. Knightly. Adding capacity points to a wireless mesh network using local search. In *INFOCOM*, 2008.

[12] S. Sen and B. Raman. Long distance wireless mesh network planning: problem formulation and solution. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 2007.